



FINAL REPORT

DEVELOPING QUALITY APPLICATIONS



CloudBeds: Hotel Booking Management System

Prof. Sarika Sultana

PREPARED BY: GROUP 06

(SEC-04)

RONAK GANDHI - 8855211

REMYA SHAJI - 893944

SHWETA PARMAR - 8947552

KASHISH RAO – 8914585

Submission Date: 20th, April 2024

Purpose	3
Project Summary	3
Project Proposal.....	3
Project Background and Description	3
Problem Statement.....	3
Project Scope	3
Exclusions from Scope	4
High-Level Requirements.....	4
Materials, Tools Required	4
High Level Design – Architecture.....	4
Purpose:	4
High Level Design:.....	5
Application Flow	7
Design Overview:	10
External Interfaces:.....	15
User Interfaces:.....	15
Other Design Considerations:	23
Database Organization and Data Storage:	24
The schema of the database tables:	24
Entity-Relationship Diagram:	27
Project Requirements	29
Purpose	29
Overall Description	29
System Features.....	30
External Interfaces Requirements	31
Non-Functional Requirements.....	31
Risk Based Test Strategy	32
Purpose	32
Traceability Matrix	32
Risks	33
Testing Priorities	34
Risk owner Roles and Responsibilities	35
Test Levels.....	36
Environmental Requirements.....	41
Acceptance Criteria.....	43
Acceptance Test.....	45

Purpose:	45
Acceptance Test Plan:.....	45
Environment Used:	45
Acceptance Test Cases:.....	46
Acceptance Test Case Results:.....	52
System Test.....	54
Purpose	54
System Test Plan	54
Environment Used	55
System Test Cases.....	55
System Testing Test Cases Results.....	64
Unit Test.....	65
Purpose:	65
Unit Test Plan.....	65
Unit Test Cases.....	65
Traceability Matrix:	73
Conclusion.....	75
Recommendation	75

Purpose

The purpose of the Hotel Booking Management System project is to develop a comprehensive platform that transforms reservation processes. This document aims to convey the preliminary design and design overview to stakeholders, facilitating understanding and aiding in identifying potential risks and challenges early in the development process. By streamlining communication and fostering collaboration, this document ensures alignment among stakeholders throughout the project lifecycle.

Project Summary

The Hotel Booking Management System project involves the design, development, and testing of a web-based application tailored for efficient booking management and customer satisfaction. It covers various phases including high-level design, unit testing, integration testing, system testing, and acceptance testing to ensure functionality, performance, and security.

Project Proposal

Project Background and Description

The Hotel Booking Management System project aims to revolutionize reservation procedures by creating an all-inclusive platform. It includes critical features such as an easy-to-use admin dashboard for effective booking management, secure user authentication, and robust reporting tools for in-depth analytics. With a prioritized focus on customer satisfaction, the project strives to optimize resource allocation, improve reporting precision, and establish a centralized hub for efficient booking management. Our iterative development process will be guided by the Agile methodology, ensuring adaptability to changing requirements and promoting continuous improvement.

Problem Statement

The current reservation systems lack comprehensive features and often exhibit issues such as complex booking processes, limited real-time availability, inefficient resource management, security concerns, and scalability challenges. To address these issues, the Hotel Booking Management System project aims to revolutionize reservations with an all-in-one solution that prioritizes user-friendly features, security, and analytics.

Project Scope

The goal of the Hotel Booking Management System project is to create a comprehensive platform that transforms reservation processes. This system will address identified pain points, ensuring a seamless booking experience for customers, real-time visibility for administrators, and a scalable architecture that can grow alongside the business. The project aims to improve reporting accuracy, optimize resource allocation, and provide a central centre for effective booking administration, with a focus on customer pleasure. The project's objectives include enhancing user experience, integrating real-time availability, ensuring scalability and security, managing user profiles, enabling booking confirmation and notifications, as well as facilitating cancellation and modification functionalities.

Project Objectives:

- Enhance User Experience
- Integrate Real-time Availability
- Ensure Scalability and Security
- Implement User Profiles

- Enable Booking Confirmation
- Facilitate Cancellation and Modification

Exclusions from Scope

- Physical property management (e.g., maintenance, housekeeping)
- Direct management of hotel staff and personnel
- Transportation services (e.g., airport shuttle, car rental)
- Food and beverage management
- Development and management of Payment and Notification System for Customers

High-Level Requirements

The high-level requirements for the Hotel Booking Management System include:

- User-friendly interface
- Secure user authentication
- Reporting and analytics
- Resource optimization
- Scalability and flexibility
- Billing and invoicing

Materials, Tools Required

The materials and tools required for the successful completion of the Hotel Booking Management System include:

- Development environment: ReactJS
- Database management system: Supabase
- Version control system: Git and GitHub
- Diagrams tools: Lucidchart, Draw.io
- Project management tool: Trello or Jira
- Testing tools for Unit Testing-PyTest
- Performance Testing-Apache JMeter
- System testing – JMeter
- Defect tracking software-MantisBT.
- Documentation and collaboration tools-Google Docs for documentation
- Communication: Microsoft Teams, or other team communication tools

High Level Design – Architecture

Purpose:

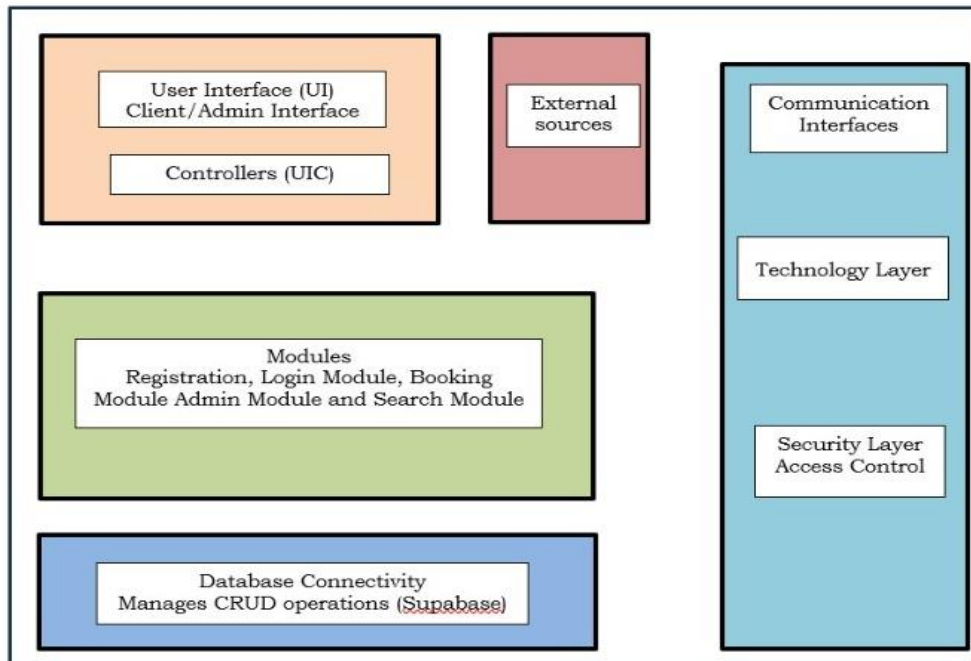
This document provides a high-level overview of the Hotel Booking Management System project architecture. It aims to convey the preliminary design and design overview to stakeholders, including

the project team and developers. The high-level design breaks down the entire system into smaller parts, facilitating understanding and aiding in identifying potential risks and challenges early in the development process. Furthermore, it aims to assist in identifying potential risks and challenges early in the development process, enabling proactive measures to mitigate these risks effectively. Through its focused and descriptive approach, this document aims to streamline communication, foster collaboration, and ensure alignment among stakeholders throughout the project lifecycle.

Requirement ID	Module_Name	Requirement Description	Test Case Description
FR1	Registration	User Registration functionality	Verify user registration functionality
FR2	Login	Login functionality	Verify user / admin login functionality
FR3	Admin(Add Post)	Create new post	Verify admin can create a new post
FR4	Search	Room Search management	Verify user room search functionality
FR5	Booking	Room Booking functionality	Verify user room booking functionality
FR6	Admin(Report)	Report analysis	Verify admin report functionality
FR7		User Profile Management functionality	Verify user profile management functionality
FR8	Admin(CRUD operation related to room)	Administrative Dashboard functionality (status updating, modify, and cancel)	Verify admin dashboard functionality
FR09	Booking(CRUD operation related to room)	Booking management functionality	Verify booking management functionality
FR10	Booking(Cancel room)	Cancellation of bookings	Verify user cancellation functionality
NFR1		Performance	Verify performance of the website
NFR2		Usability	Verify usability features
NFR3		Compatibility	Verify compatibility on any platform
NFR4		Authentication and Security	Verify security features implementation

High Level Design:

APPLICATION ARCHITECTURE



❖ Presentation Layer

- The application's graphical elements that users interact with are represented by this layer. Screens, forms, buttons, and other graphical elements are all part of it.
- The fact that only one Admin Interface is described suggests that the user interface was created with administrative duties in mind.

❖ Business Logic Layer Modules

- This layer encapsulates the application's business logic and functional units, ensuring a modular and organized architecture.
- Key modules include:
 - User Management: Responsible for user authentication, registration, and profile management functionalities. It handles user interactions related to login, registration, and profile updates.
 - Booking Management: Facilitates room reservation, cancellation, and modification functionalities. This module handles the core booking processes, ensuring smooth and efficient management of room bookings.
 - Admin Dashboard: Provides administrative functionalities for managing users, bookings, and other system configurations. It offers features such as user role management, booking analytics, and system settings customization.
 - Search Module: Enables users to search for available rooms based on their preferences, such as date range, room type, and amenities. It retrieves relevant room information from the database and presents it to the user for selection.

❖ Database Connectivity:

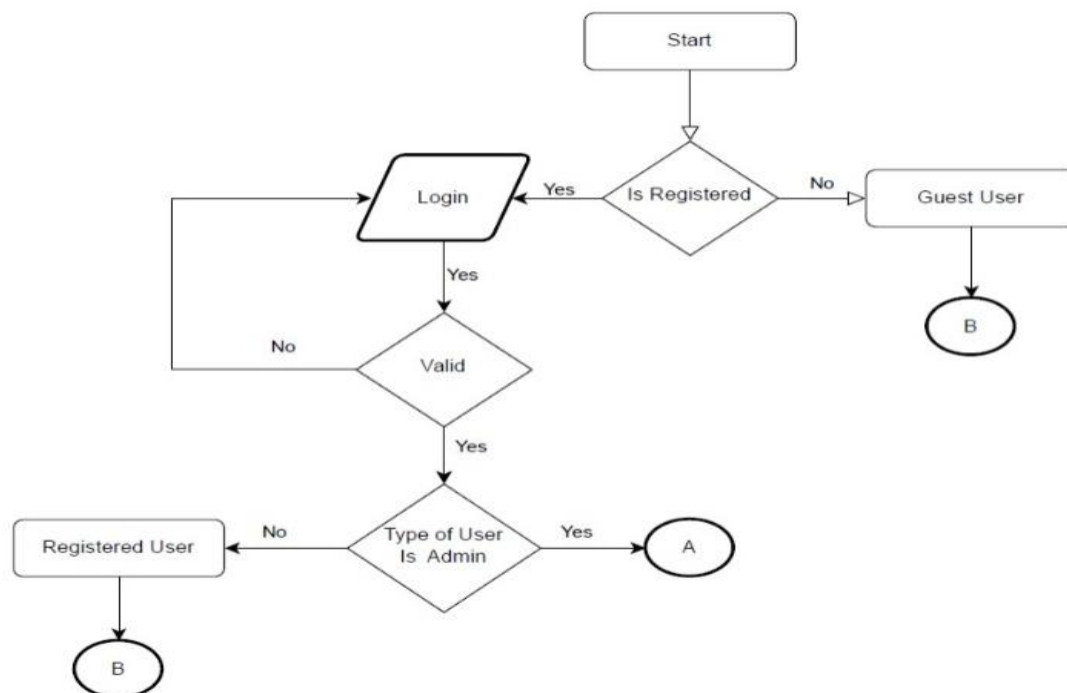
- This layer interacts with the database management system (DBMS) to perform CRUD operations (Create, Read, Update, Delete) and manage data persistence.
- Supabase is designated as the primary database management system (DBMS), indicating that the application utilizes Supabase for storing and retrieving data related to users, bookings, rooms, transactions, etc.
- Supabase offers a scalable and flexible cloud-based solution for data storage, providing reliability and efficiency in managing the Hotel Booking Management System's data requirements.

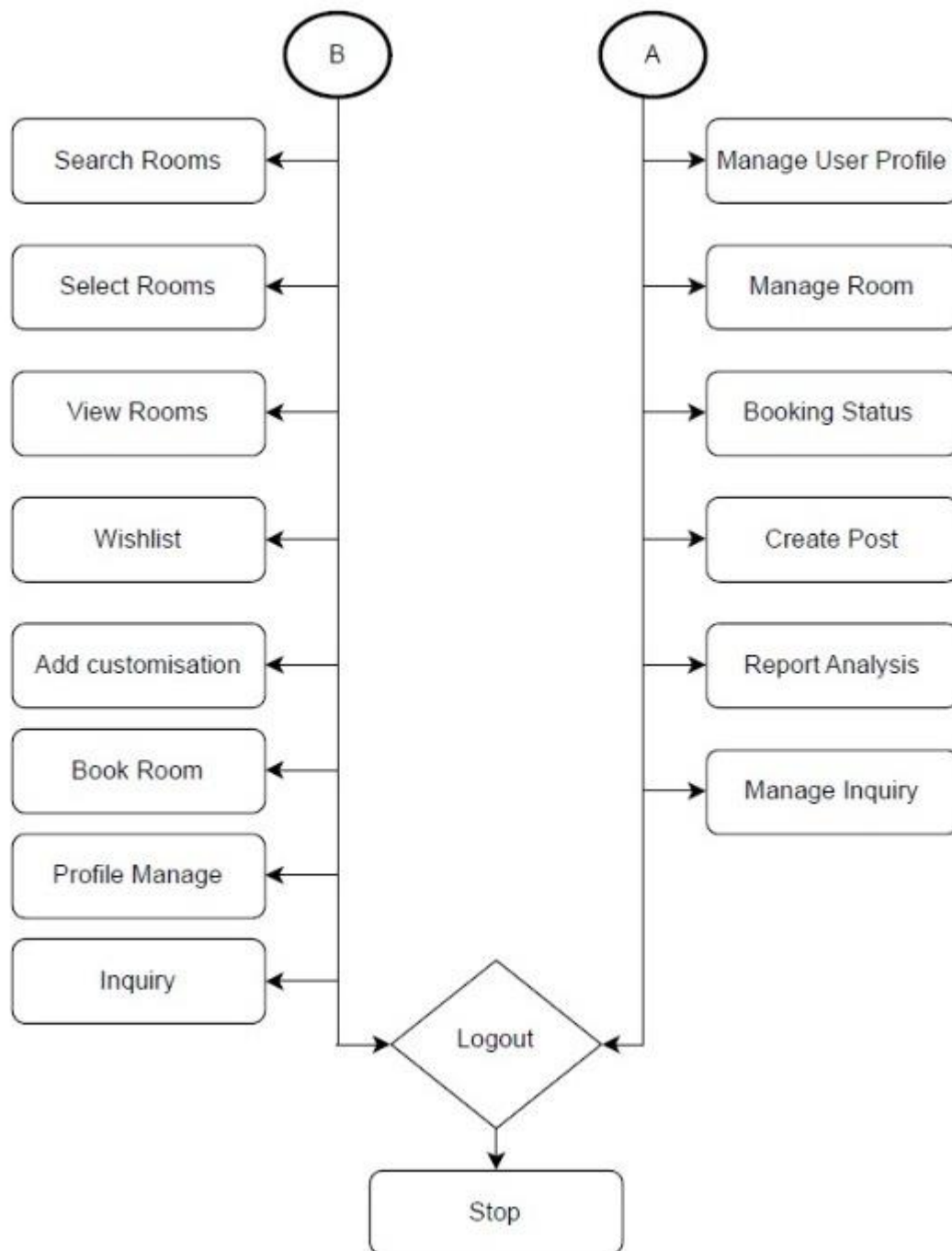
❖ Technology Layer

Operating System:

- The technology layer encompasses the underlying software and tools used in the development process.
- Technologies such as React.js for front-end development, Node.js for server-side scripting, and Express.js for building RESTful APIs are utilized.
- Supabase is employed as the primary NoSQL database for flexible and scalable data storage.

Application Flow





❖ Preliminary Design:

The preliminary design of the Hotel Booking Management System aims to lay the groundwork for the project by addressing key aspects such as sizing, risk assessment, and identification of critical components.

Sizing the Project:

This project aims to be a game-changer for hotel reservations. To achieve this, we need to thoroughly assess the project's size and intricacies. This in-depth analysis will help us figure out the people and the effort needed to build the system. By clearly understanding the project's

scale, stakeholders can make informed decisions about resource allocation and set achievable deadlines.

Risk Assessment:

One of the key goals of the preliminary design phase is to identify potential risks and obstacles that may arise throughout the development process. In the instance of the Hotel Booking Management System, implementing security measures and efficiently building the booking module have been highlighted as potentially risky or time-consuming tasks. By proactively managing these risks, the project can avoid any delays and maintain smooth progress.

Identification of Critical Components:

The preliminary design also includes the identification of the system's important components and subsystems. The Hotel Booking Management System's primary components include user identification, booking management, reporting tools, and scalability. Developers can better prioritize development efforts and allocate resources by breaking down the system into smaller, more manageable sections.

Design Overview:

The document provides a detailed knowledge of the Hotel Booking Management System, including its dependencies, stakeholders, and design considerations. By giving a comprehensive picture of the system's components and operations, it enables project teams to collaborate and make better decisions.

System Architecture:

The system architecture is tailored to meet the specific needs of hotel booking management. It has several layers, including presentation, application logic, and data storage. This design provides a scalable and durable basis that can meet the complex needs of the hotel industry.

Addressing Pain Points:

Identifying and resolving frequent pain points in hotel reservation procedures is crucial for effective system design. The technology seeks to ease these pain points by prioritizing features like faster booking experiences for customers and real-time visibility for administrators.

Scalability and Growth:

The system architecture is built for scalability and growth, taking into account the hospitality industry's dynamic character. It can effortlessly adjust to meet the growing demands of hotel bookings, rising user numbers, and increased data needs. This scalability means that the Hotel Booking Management System is adaptable and responsive to changing business requirements throughout time.

Reporting and Analytics:

The system offers powerful reporting and analytics features to provide administrators with actionable insights. These capabilities allow hotel management to get useful information about booking trends, customer preferences, and resource utilization. Hotels that use data-driven

decision-making can optimize resource allocation, improve operational efficiency, and increase overall visitor happiness.

Customer Satisfaction:

Our system design prioritizes client satisfaction. Every component of the system, from user interfaces to booking management features, is methodically planned to exceed consumer expectations. The Hotel Booking Management System strives to create long-term guest loyalty and pleasure by offering intuitive interfaces, personalized booking experiences, and effective administrative tools.

Design Overview:

A high-level design gives an overview of a system, product, service, or process while assuring interoperability with supporting components. This section is to detail the design considerations and dependencies of the Hotel Booking Management System.

Dependencies:

The project relies on platforms, systems, products, services, and processes such as ReactJS for front-end development, Supabase for backend database management, and various third-party APIs for additional functionalities.

Stakeholders:

The project's key stakeholders include project sponsors, product owners, end users (customers), development team members, and administrative staff. Each stakeholder has distinct duties and responsibilities, which provide effective collaboration and communication throughout the project's lifecycle.

Design Considerations:

Memory management, CPU utilisation, error handling, and security measures are all important design considerations. Measures are in place to address potential risks, concerns, and assumptions in the commercial, legal, environmental, security, safety, and technical areas.

End User Experience:

The design prioritises end users' wants and expectations in order to provide a seamless and intuitive user experience. Customer happiness and loyalty are improved by integrating features such as user-friendly interfaces, simpler booking processes, and responsive customer care.

Architecture Diagram: Create a high-level block diagram that shows the main components of your system interacting with each other. Examples include:

User Interface (UI) - handles user interactions

Application Layer - processes user requests and interacts with the database

Database - stores system data

Admin Panel (optional) - manages users, bookings, etc.

Component Descriptions: Briefly describe the function of each component in the diagram.

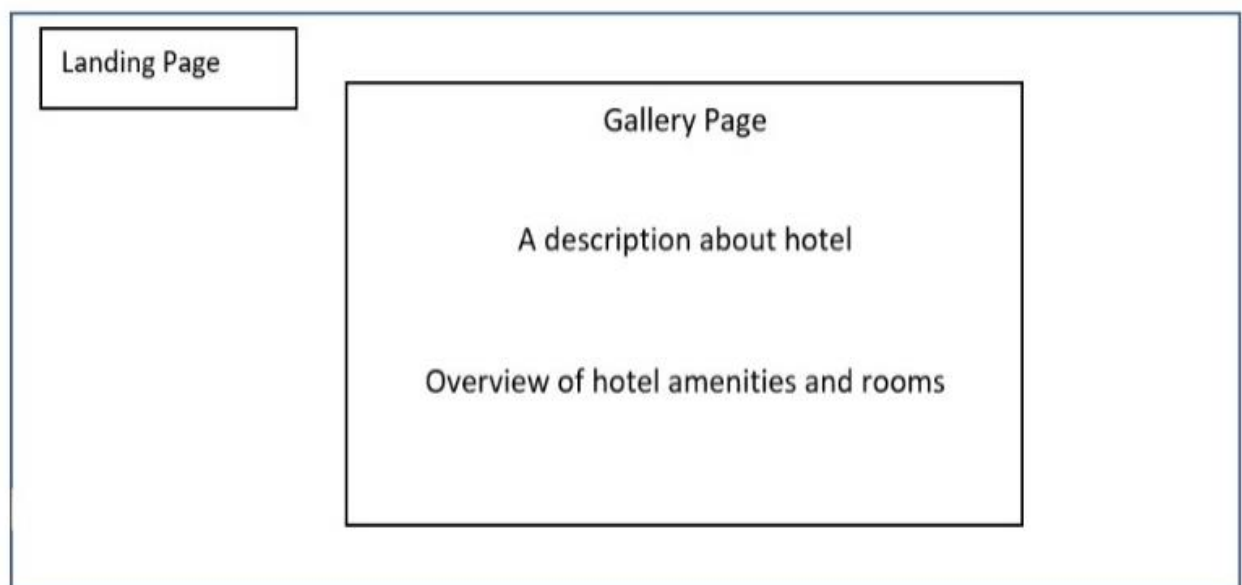
Technology Stack: Mention the chosen technologies for:

Application layer (e.g., programming language, framework)

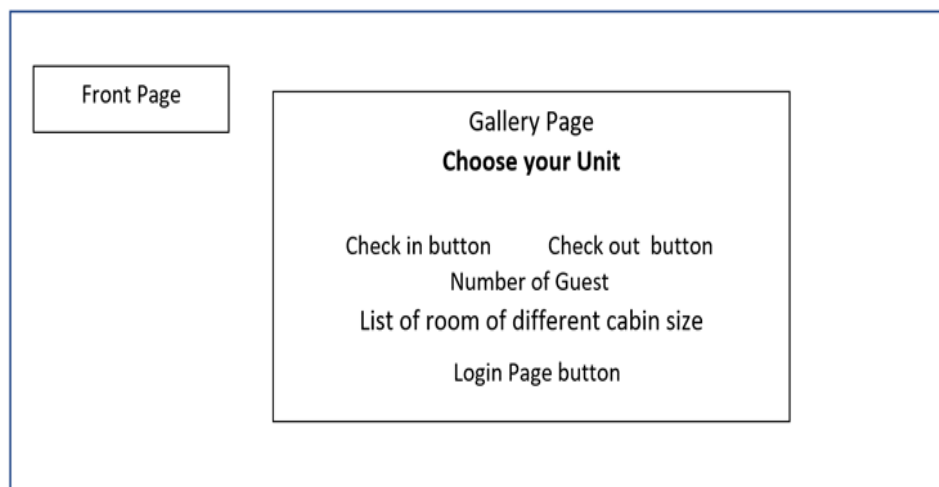
Database Management System (DBMS)

Design Overview: Design overview contains the overall flow of the system along with the structured design.

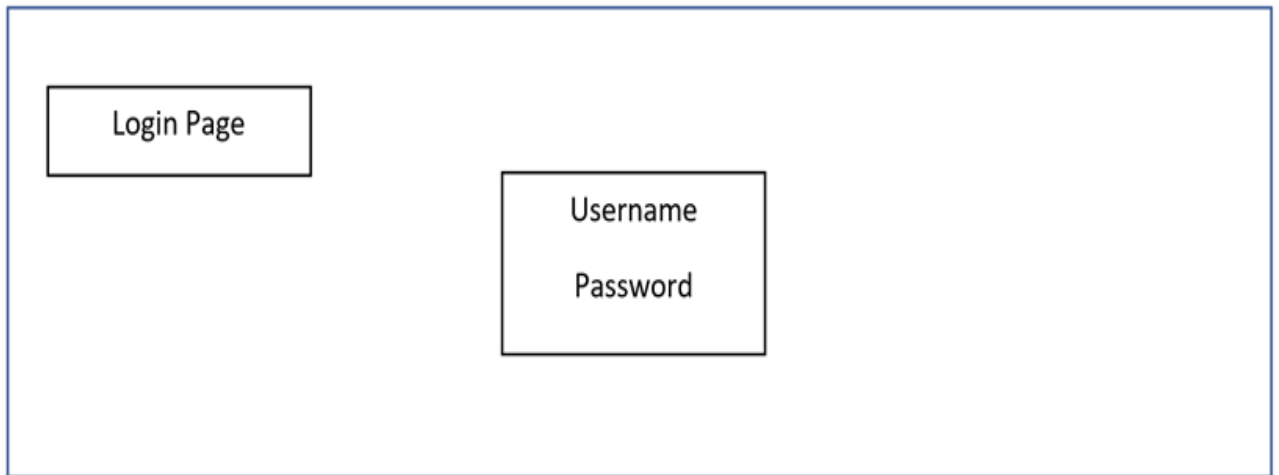
(1) Landing Page:



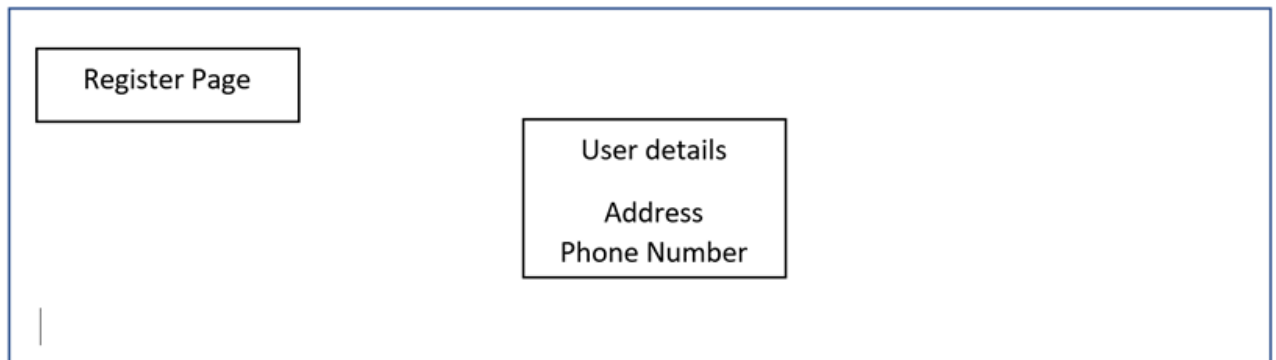
(2) Front Page:



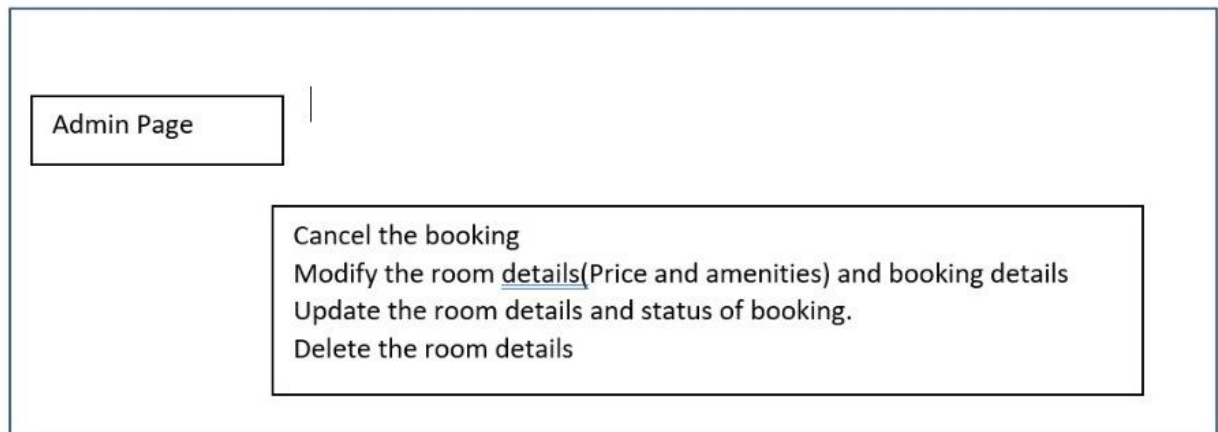
(3) Login Page:



(4) Registration Page:



(5) Admin Page



(6) Booking page:

Booking Page after selecting the room

Highlights of [room details](#)

Reviews of Previous users

Space details

Unit Amenities

Food and drink

Internet

Extra facility

Cancellation

Policies

Non-Refundable

Fully Refundable before 15 days

Fully Refundable After 5 days

Fully Refundable

Extras:

no extra

breakfast

bed

taxi service

Estimated total price including tax
price [details](#)

Reserve button

(5) Reserve Page:

Reserve Page

1. Pay now
2. Pay when you stay

Who is checking in details:

Name , email address and phone number
Option to : receive alert mails, customize any special request

Payment details: Name on card,
debit/credit card number
expiry date , security code

(6) Confirmation Page:

Confirmation

Customized message with summary details of your boooking

Use-Case Diagram:

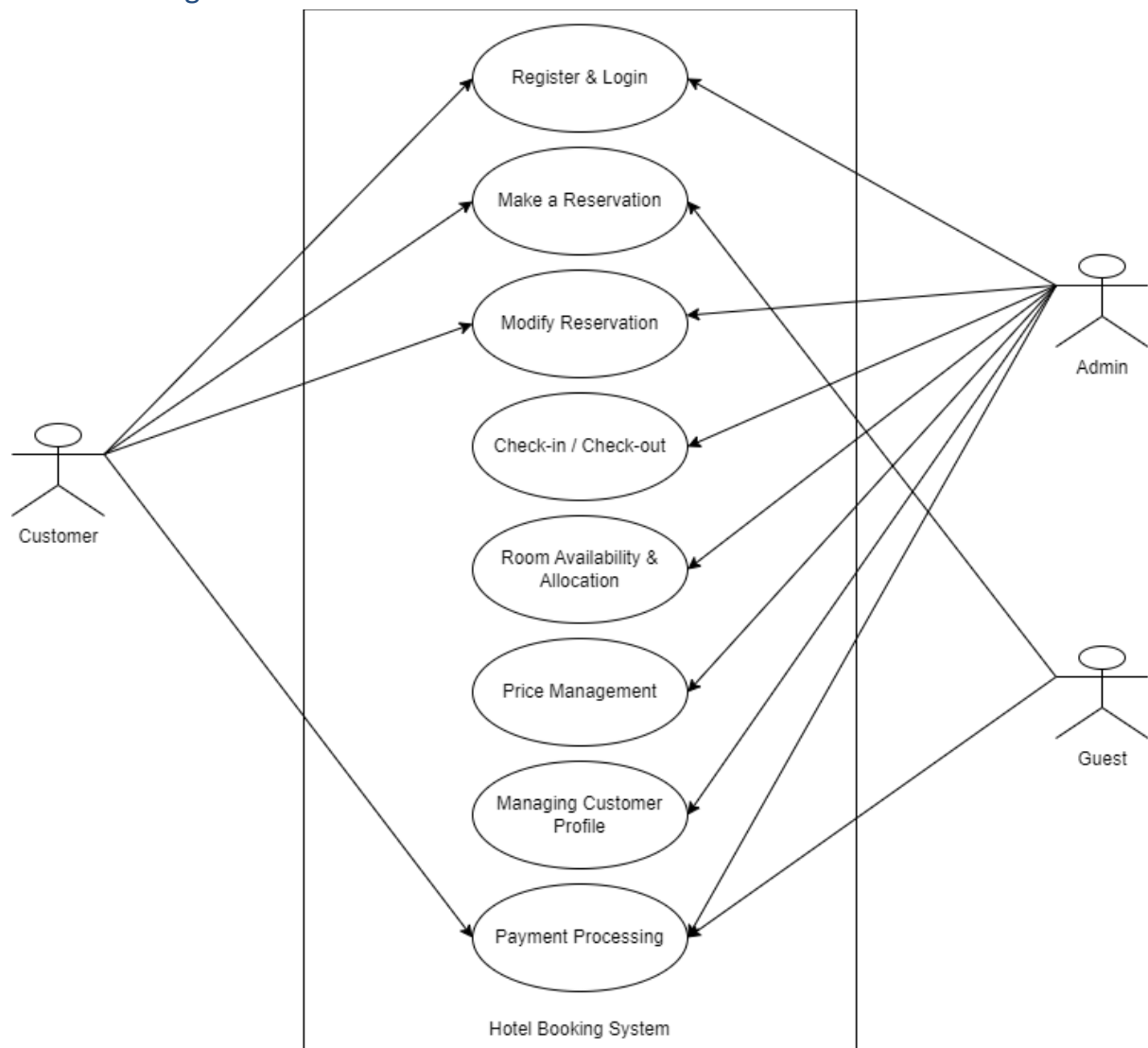


Figure 1: Use Case Diagram

External Interfaces:

External interfaces include communication channels with third-party services such as payment gateways, email services for notifications, and API integrations for hotel data retrieval. Requirements for user interfaces, hardware interfaces, and software interfaces are outlined to ensure seamless interaction with external systems.

User Interfaces:

The user interfaces of the Hotel Booking Management System are designed to provide an intuitive and seamless experience for users. Users can browse the website using their personal devices and easily navigate through various options such as selecting their stay, choosing a room, and adding additional facilities. Screenshots of the application interface, showcasing the menu structure, screen layouts, and input mechanisms, are provided in workflow design.

The user interfaces encompass various screens and menus accessible to users, including registration forms, login screens, search filters, booking forms, profile management pages, and administrative dashboards. Menu structures, screen layouts, and report formats are defined to enhance user experience and usability.

Menu Structure:

- Home: Navigates users to the homepage featuring carousel images of the hotel and its amenities.
- About Us: Provides information about the hotel, its history, and mission.
- Room Showcase: Displays details of different rooms available for booking, including prices, amenities, and policies.
- Book Room: Allows users to browse available rooms, apply filters, and proceed with booking.
- My Bookings: Appears for logged-in users and provides access to their booking history.
- Admin Dashboard: Exclusive access for administrators to manage bookings and view recent activity.
- Admin Menu Dropdown:
 - Cabin Details: Displays detailed information about each room type, including occupancy, amenities, and pricing.
 - Booking Details: Provides an overview of all bookings made by users.

Screen Names and Purposes:

(1) Home Page with Carousel:

- Features carousel images showcasing the hotel's amenities and a login button for user authentication.
- Includes options to navigate to other sections like about us, room showcase, and booking.

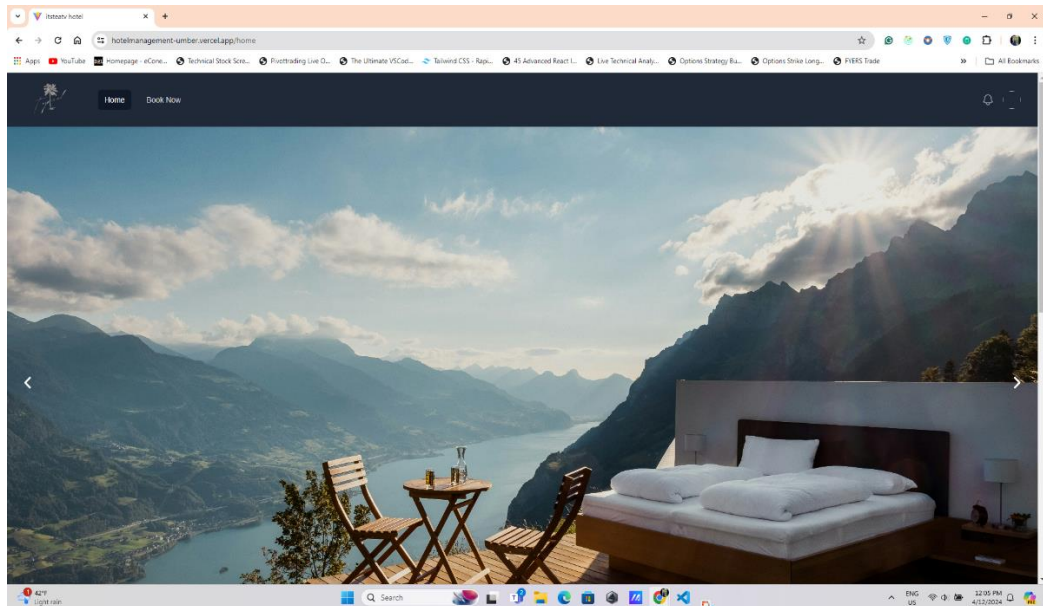


Figure 1: Home Page with Carousel

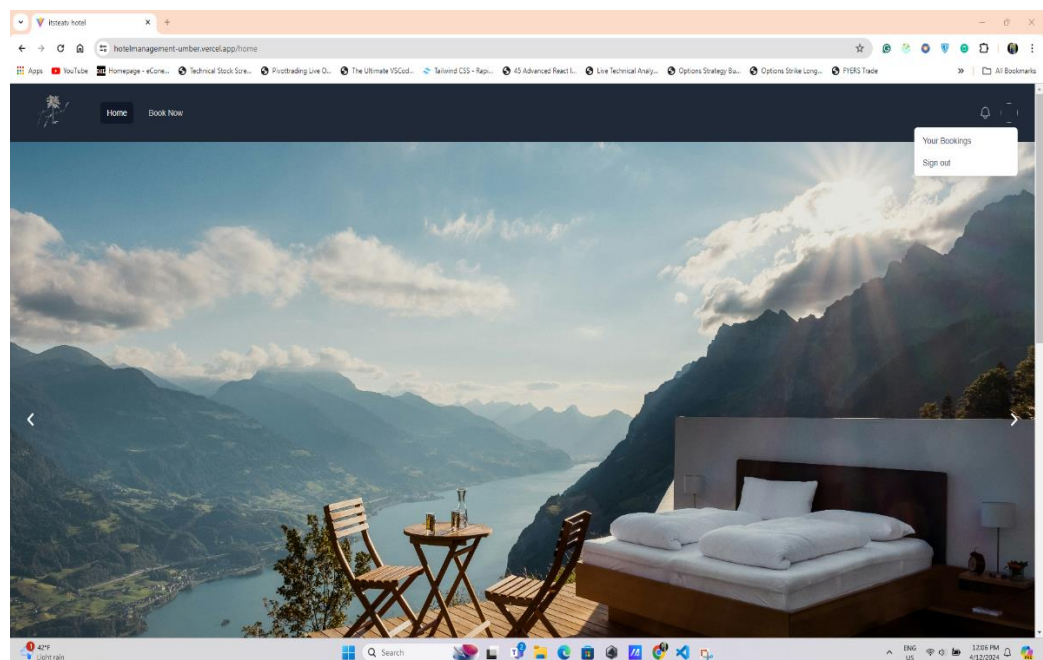


Figure 2: Home Page with Carousel and DropDown menu

(2) About Us and Room Showcase:

- Provides information about the hotel and showcases details of different rooms, including prices, amenities, and policies.

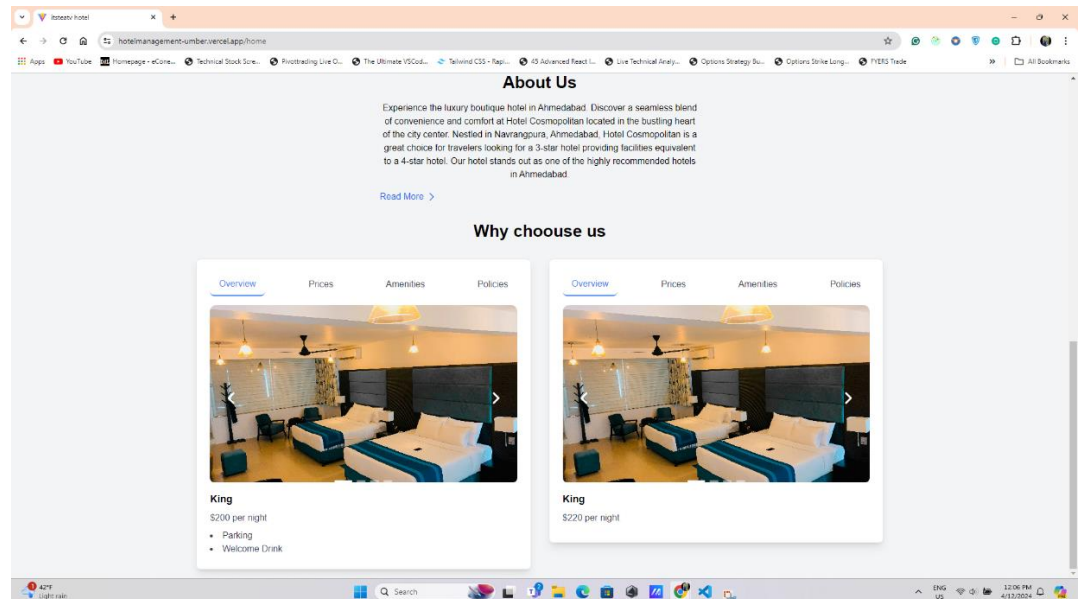


Figure 3: About us and Room showcase

(3) Book Room Page with Filters:

- Displays all available rooms with filtering options based on check-in/check-out dates and the number of guests.

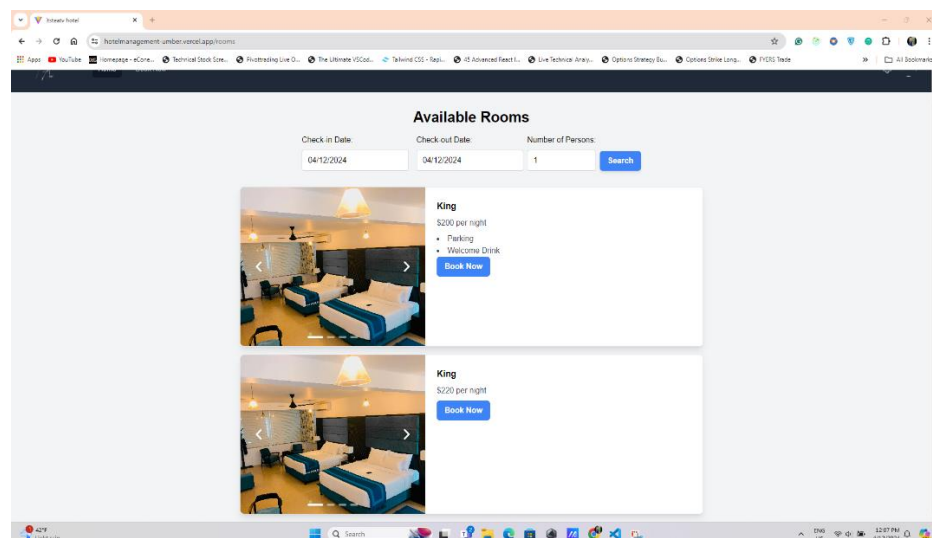


Figure 4: Book Room Page with filters

(4) Booking Form Autofilled if Logged in:

- Autofills booking details for logged-in users, streamlining the booking process.

The screenshot shows a web browser window with the URL `localhost:3000/rooms`. The page displays a "Booking Room" form with the following fields and values:

- First name: Ronak
- Last name: Gandhi
- Email: ronak@gmail.com
- Date of Birth: 04/12/2024
- Phone Number: 9999999999
- Address: Eliza
- Number of Adults: 1
- Number of Children: 0
- Room Add Ons: Room Add Ons
- Extra Services: Room Add Ons
- Price:
 - Room : 200 for 1 days
 - Room Add Ons : 0 for 1 days
 - Extra Add Ons : 0 for 1 days
 - Total Price : 200

At the bottom right of the form, there are "Cancel" and "Book" buttons.

Figure 5: Booking Form Auto filled if logged in.

(5) My Bookings:

- Shows the booking history for logged-in users, providing access to view, modify, or cancel bookings.

The screenshot shows a web browser window with the URL `localhost:3000/mybookings`. The page displays a "My Bookings" section with a table of booking history:

CABIN	GUEST	DATES	STATUS	AMOUNT	ACTIONS
King	Ronak Gandhi ronak@gmail.com	12 hours ago -- 0 night stay Apr 19 2024 -- Apr 19 2024	Booked	\$200.00	i
King	Ronak Gandhi ronak@gmail.com	12 hours ago -- 0 night stay Apr 19 2024 -- Apr 19 2024	Booked	\$200.00	i

Figure 6: Booking Information

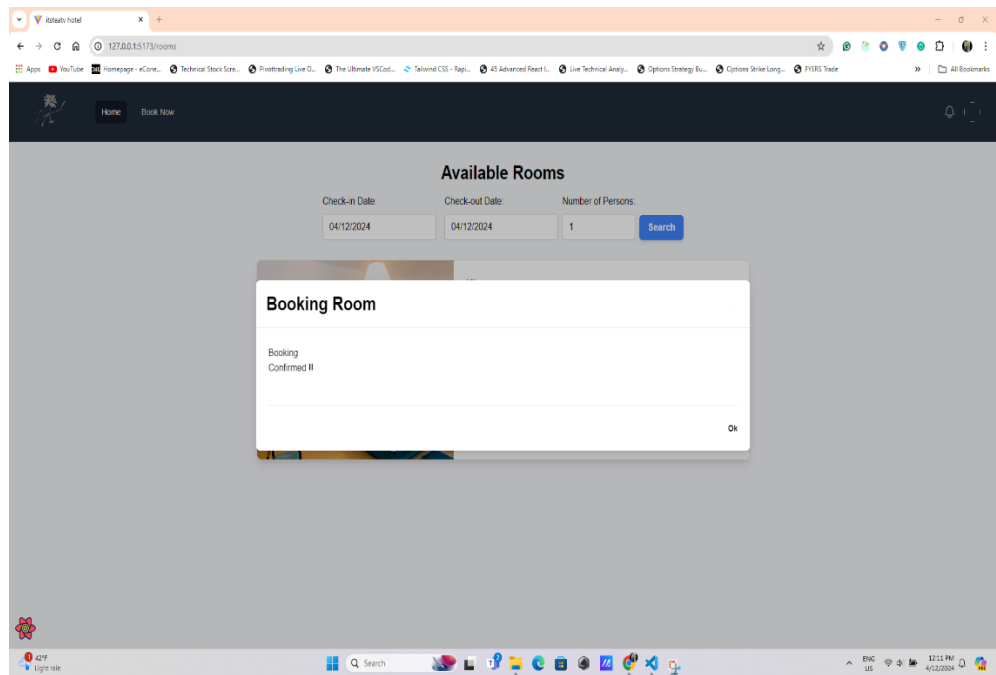


Figure 7: Booking Confirmed

(6) Single Booking Information:

- Displays a summary of the booking details after confirmation, including room type, dates, and total cost.

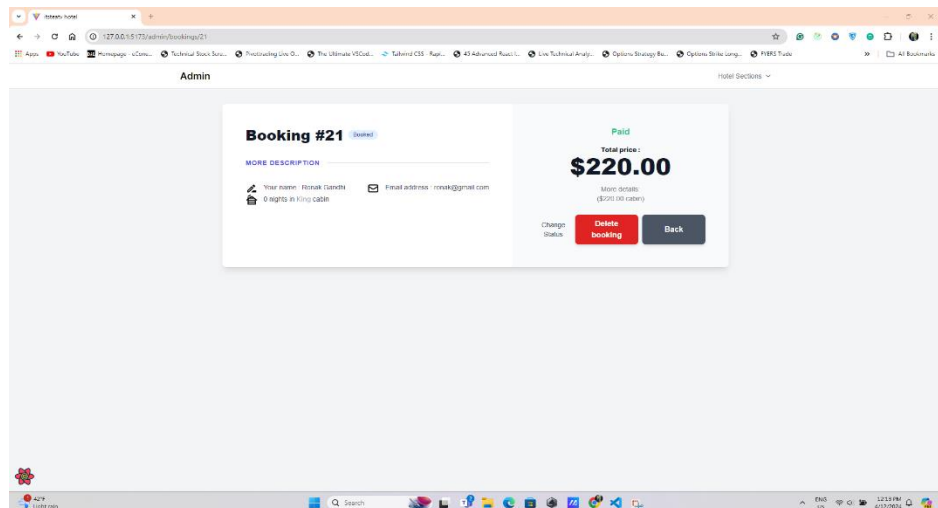


Figure 8: Single Booking Information

(7) Admin Dashboard:

- Offers administrators a centralized platform to manage bookings, view recent activity, and access essential functionalities.

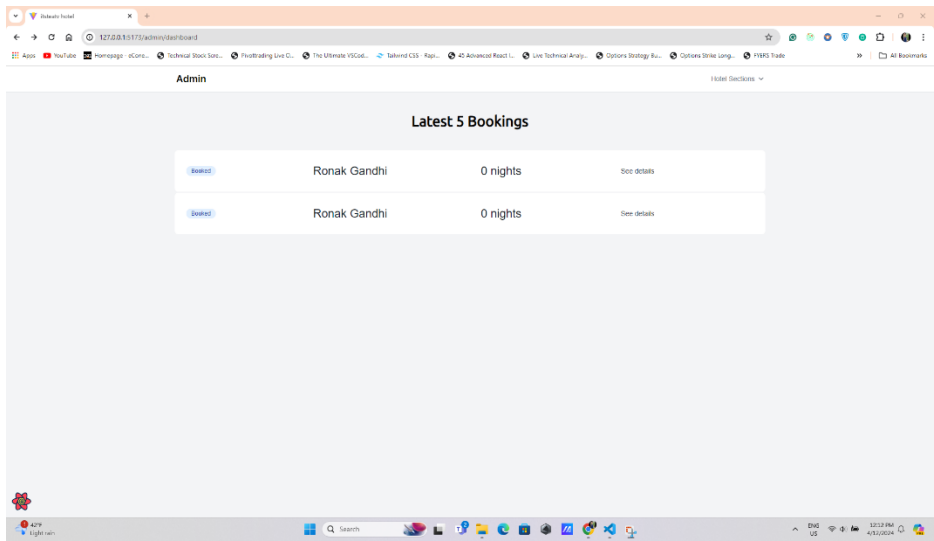


Figure 9: Admin Dashboard

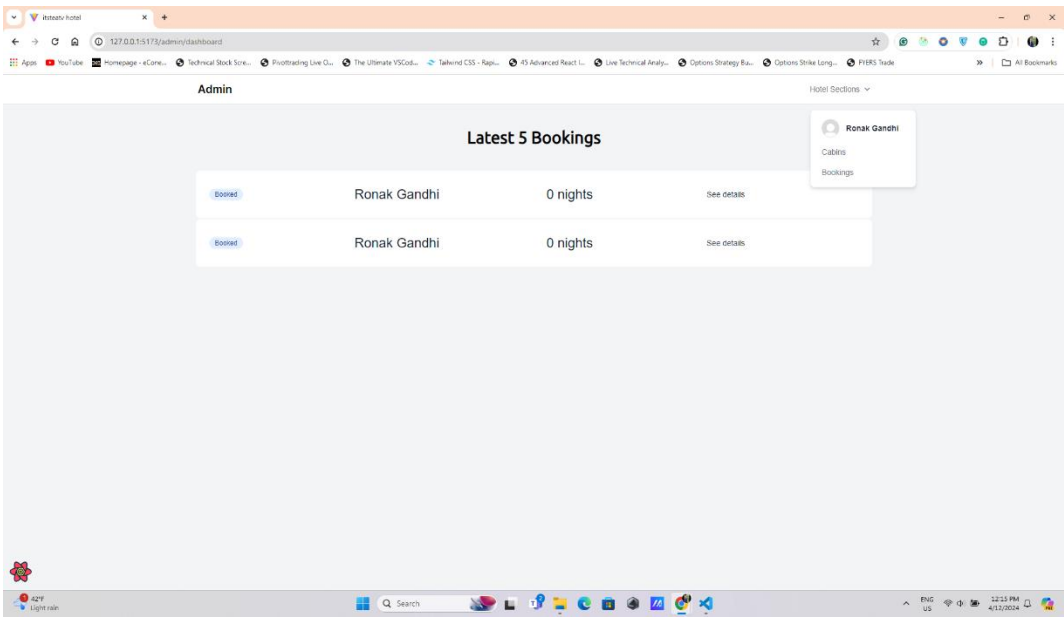


Figure 10: Admin Dashboard Menu dropdown

(8) Delete Confirmation Popup:

- Appears when an administrator attempts to delete a booking, ensuring confirmation before proceeding.

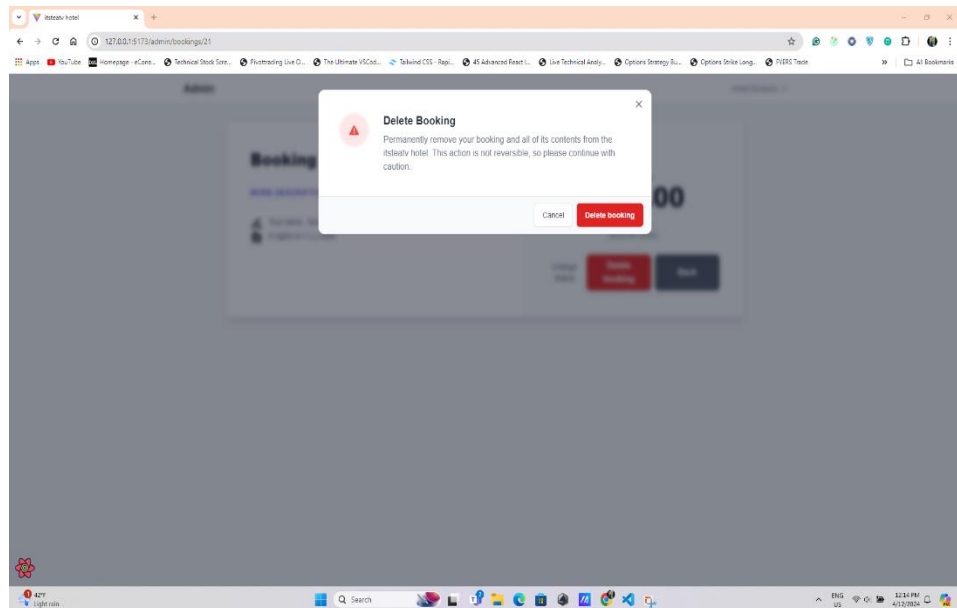


Figure 11: Delete Confirmation Popup

(9) Bookings Action Button:

- Provides options for administrators to take action on bookings, such as modifying or canceling them.

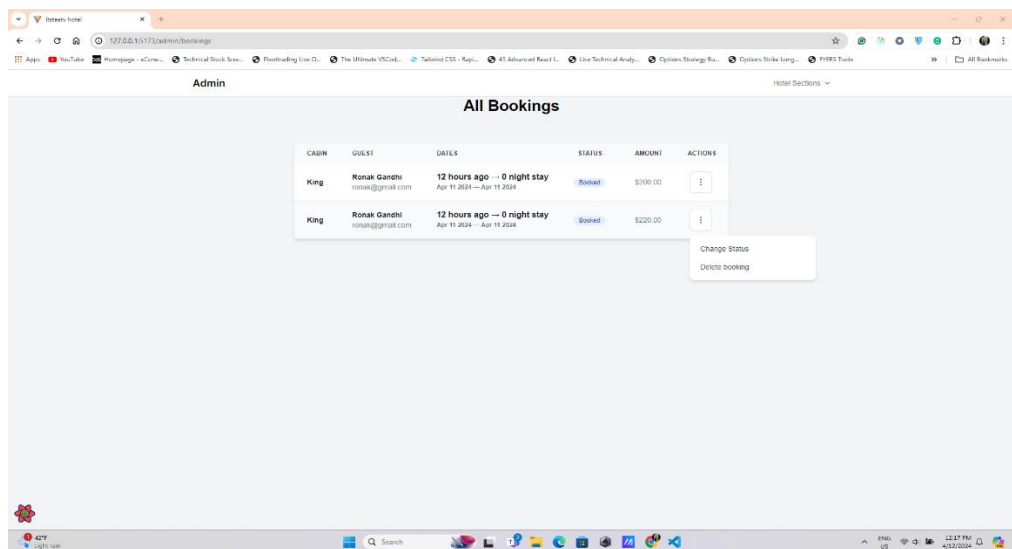


Figure 12: Bookings Action Button

(10) Change Status Dropdown:

- Allows administrators to change the status of bookings, such as confirming or canceling them.

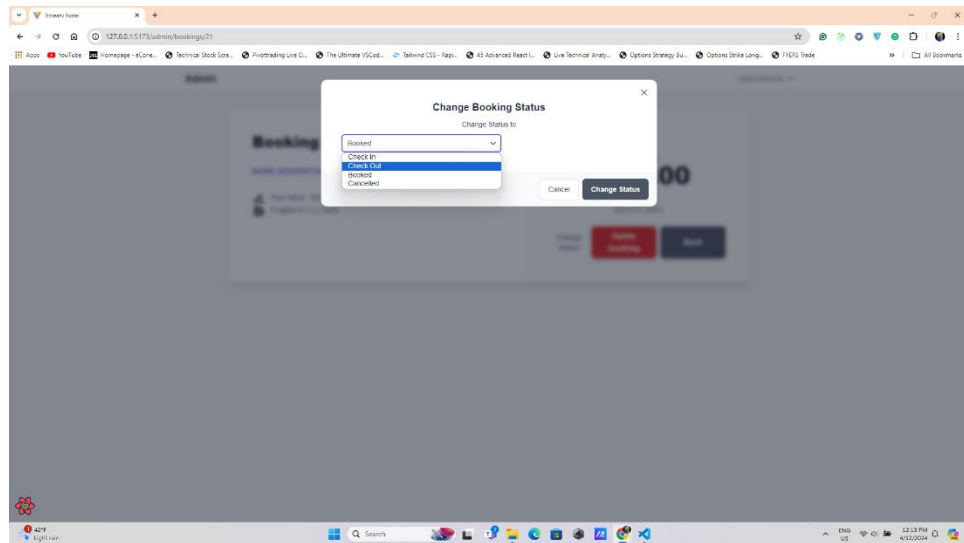


Figure 13: Change Status Dropdown

Other Design Considerations:

Device Compatibility:

The system should be designed to be responsive and compatible with various devices, including desktops, laptops, tablets, and smartphones, ensuring a seamless browsing experience for users across different platforms.

Room Selection and Add-On Facilities:

Users should be able to browse available rooms, view room details, and select their preferred stay duration. Additionally, the system should allow users to add extra facilities such as meals, spa services, and transportation options to their booking.

Payment Methods and Confirmation:

Support for online payment methods such as credit cards and debit cards should be implemented securely to facilitate instant transactions. Offline payment options should also be available for users who prefer alternative payment methods. Upon successful payment, users should receive a confirmation message via email and SMS, ensuring transparency and reliability in the booking process.

Booking Tracking and Customization:

Users should have the ability to track the status of their bookings in real-time, providing them with updates on reservation details and any modifications made. Additionally, customization features should be available to users, allowing them to personalize their bookings according to their preferences.

Alerts and Reminders:

An alert system should be integrated to notify users about important updates, such as booking confirmations, payment reminders, and upcoming reservations. These alerts can be sent via email and SMS to ensure timely communication with users.

Constraints:

External costs associated with payment processing, add-on services such as taxi bookings, and hosting on web services should be considered as constraints. The system should be designed to optimize resource utilization and minimize operational costs while providing value-added services to users. Additionally, data privacy and security measures should be implemented to protect user information and ensure compliance with regulatory requirements.

Database Organization and Data Storage:

The schema of the database tables:

payment:
paymentId: int8 (primary key)
amount: varchar
method: text

reservation:
reservationId: int8 (primary key)
customerId: int8 (foreign key referencing customer)
roomId: int8 (foreign key referencing room)
adminId: int8 (foreign key referencing admin)
checkInDate: timestampz
checkOutDate: timestampz
totalCost: float8
status: bool
numberOfRooms: int8
numberOfPeople: int8
numberOfChildren: int8
desositAmount: float8
paymentId: int8 (foreign key referencing payment)

room:
roomId: int8 (primary key)
roomNumber: int8

roomTypeId: int8 (foreign key referencing room type)
pricePerNight: float8

amenities:
amenitiesId: int8 (primary key)
name: varchar

roomAmenities:
roomTypeId: int8 (primary key) (foreign key referencing roomType)
amenitiesId: int8 (primary key) (foreign key referencing amenities)

roomAdd-ons:
roomTypeId: int8 (primary key) (foreign key referencing roomType)
add-onId: int8 (primary key) (foreign key referencing add-on)

admin:
adminId: int8 (primary key)
username: varchar
password: varchar

room type:
roomTypeId: int8 (primary key)
roomTypeName: varchar
description: varchar
maxOccupancy: varchar

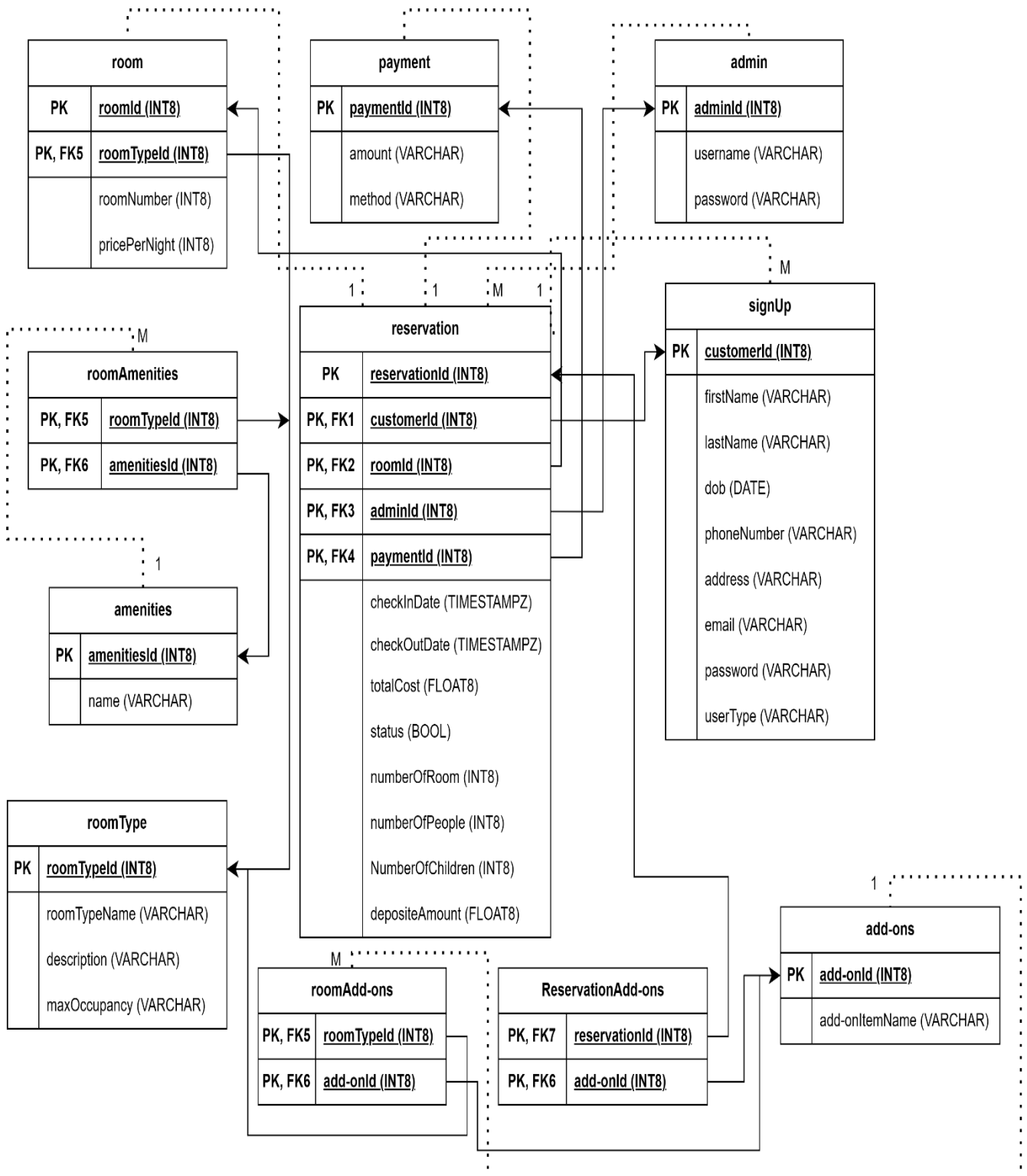
reservationAdd-ons:
reservationId: int8 (primary key) (foreign key referencing reservation)
add-onId: int8 (primary key) (foreign key referencing add-on)

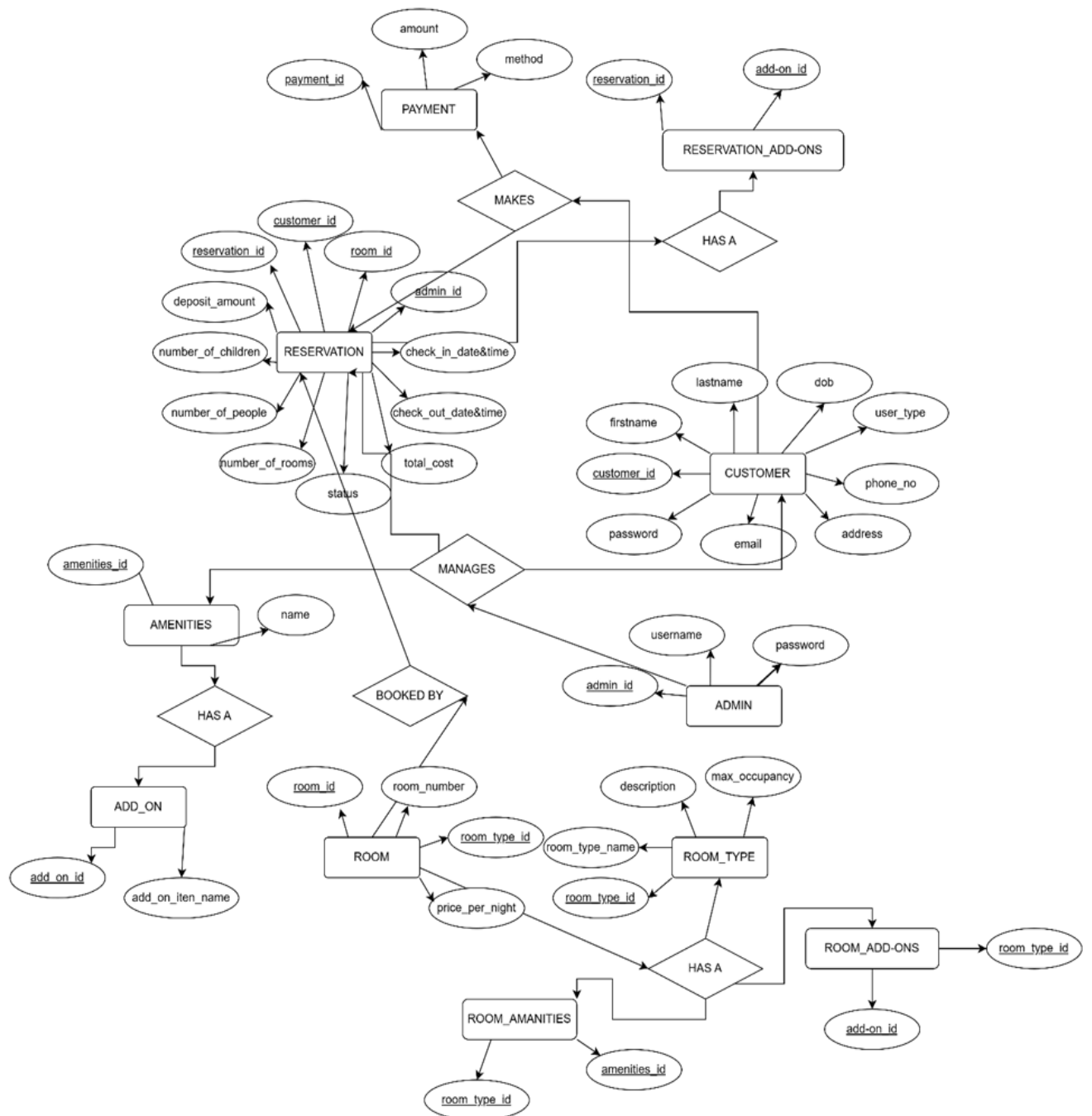
add-on:
add-onId: int8 (primary key)
add-onItemName: varchar (unique)

customer: (signUp)
customerId: int8 (primary key)
firstName: text

lastName: text
dob: date
phoneNumber: text
address: text
email: text
password: text
userType: text

Entity-Relationship Diagram:





Project Requirements

Purpose

This document delineates the functional and non-functional requirements for the Hotel Booking Management System project architecture. It aims to provide a clear understanding of the project's scope, expectations, functionalities, and limitations to stakeholders, including developers, project managers, and users. By establishing alignment among stakeholders, the document facilitates effective collaboration towards achieving the project's objectives. With defined requirements and constraints, stakeholders can work cohesively to ensure the successful development and implementation of the Hotel Booking Management System, thereby enhancing reservation processes and customer satisfaction.

Overall Description

Hotel Booking Management System is designed as a comprehensive web-based platform, aiming to streamline and enhance the booking experience for both customers and administrators. Key features include user registration, login, password management, profile management for both customers and businesses, booking processing, and robust search functionalities for rooms. The system's primary objective is to improve the efficiency and competitiveness of hotels in the digital realm, adapting to the evolving expectations of customers in the contemporary era of online reservation services.

Product Perspectives

The Hotel Booking Management System employs ReactJS for the front-end, Supabase for database management, and serverless technologies for scalability. This tech stack ensures a dynamic user interface, seamless data management, and flexibility in handling varying workloads, aligning with the project's goals of enhancing user experience and resource utilization.

Product Features

- Room Management

User Roles and Characteristics

- Admin: Manage Bookings, Add New Rooms
- Customer: Register, Login, Search for Rooms, Book Rooms

Operating Environment(s)

Runs on Windows 11 with ReactJS as the scripting language and SupaBase for database management.

Constraints

- Relies on ReactJS, SupaBase.
- User's internet connectivity.

Assumptions

- Users are familiar with the fundamentals of using online apps.
- For functioning, constant internet connectivity is required.

Risks

- Security of data and privacy issues.
- Relies on consistent server performance.

Dependencies

- JavaScript for server-side scripting.
- SupaBase for database storage.

System Features

User story 1:

As a new user, I want to manage my reservations and preferences by creating a user account on the hotel booking platform.

Acceptance criteria:

- i. There should be an easy-to-use form on the registration page that asks for my full name, email address, password, and any other information that may be required.
- ii. The system ought to verify that the email address entered is distinct and hasn't already been registered with it.
- iii. To make sure it satisfies security needs, password validation should be implemented.
- iv. I should get an email confirming my registration and a link to confirm my email address after submitting the form successfully.
- v. My account should be recognized as validated after accepting the link for verification in the email, and I should be taken to the login page.

User story 2:

I want to be able to quickly reset my password as a user who has forgotten it.

Acceptance criteria:

- i. The login page ought to have a "Forgot Password" tab that takes me to a place where I may reset my password.
- ii. I should be asked to provide my enrolled email address on the password reset page.
- iii. I should get a notification with a link that allows me to reset my password after entering my email address.
- iv. I should be redirected to a website where I can input a new password after selecting the reset password link.
- v. Before enabling me to set the new password, the system ought to verify it against security specifications.
- vi. My password should be successfully reset, and I should get a confirmation email.

User story 3:

I would like to have the ability to reserve a room online for the dates I wish to visit as a guest.

Acceptance Criteria:

- i. The interface on the system for choosing the dates of check-in and checkout should be easy to use.
- ii. Based on the dates you have chosen, the system ought to offer a selection of rooms that are available.
- iii. It should be readily accessible to the user to opt for a room type and finish the reservation.

- iv. The entire cost, with taxes and any other costs, should be computed and shown by the system.
- v. The system ought to provide the guest with a confirmation email with booking information as soon as the reservation is validated.

User story 4:

I would prefer to have the option of able to change or cancel my reservation as a guest.

Acceptance Criteria:

- i. The guest should be able access to their bookings by logging into their account on the system.
- ii. It should be possible for the visitor to see their future reservations.
- iii. The system must offer choices for changing the reservation dates or cancelling the reservation completely.
- iv. When making changes to a reservation, the system ought to verify if the new dates are available and adjust the reservation appropriately.
- v. The system ought to provide the guest with a confirmation email upon update or cancellation.

External Interfaces Requirements

- User Interface: ReactJs
- Software Interface: Postgres, JavaScript, Database: Supabase,

Non-Functional Requirements

- Scalability: The ability to effectively handle growing loads and client data.
- Usability: Simple feature access and intuitive navigation.
- Compatibility: Support for a range of browsers and devices.
- Performance: JavaScript that have been optimized for quick response times.
- Software quality includes a user-friendly interface, cross-platform compatibility, and dependable functioning.

Risk Based Test Strategy

Purpose

This document aims to articulate the overall test strategy for a hotel booking management system project. This system will encompass a comprehensive suite of functionalities tailored to streamline the booking process for users, ensuring flexibility and supporting diverse scenarios related to room reservations and stays. Built using React.js code, the project will prioritize smooth user registration processes, robust administrative functionality, and user-friendly features to improve the booking management experience. This document will provide the test strategy that will steer the project roadmap, ensuring that testing activities are tightly aligned with the objectives and complexities of the developing system.

Traceability Matrix

Requirement ID	Module_Name	Requirement Description	Test Case Description
FR1	Registration	User Registration functionality	Verify user registration functionality
FR2	Login	Login functionality	Verify user / admin login functionality
FR3	Admin (Add Post)	Create new post	Verify admin can create a new post
FR4	Search	Room Search management	Verify user room search functionality
FR5	Booking	Room Booking functionality	Verify user room booking functionality
FR6	Admin (Report)	Report analysis	Verify admin report functionality
FR7		User Profile Management functionality	Verify user profile management functionality
FR8	Admin (CRUD operation related to room)	Administrative Dashboard functionality (status updating, modify, and cancel)	Verify admin dashboard functionality
FR09	Booking (CRUD operation related to room)	Booking management functionality	Verify booking management functionality

FR10	Booking (Cancel room)	Cancellation of bookings	Verify user cancellation functionality
NFR1		Performance	Verify performance of the website
NFR2		Usability	Verify usability features
NFR3		Compatibility	Verify compatibility on any platform

Risks

Constraints

- Technology Restrictions: Using technologies like Visual Studio Code, Supabase, and React.js may result in limitations on how well they integrate, scale, and work with current systems.
- Resource Constraints: The project's timing and testing and development quality may be impacted by the scarcity of experienced developers, testers, and other resources, such as tools and technologies.
- Time Restraints: Time restrictions may arise when using Agile approach, particularly when there are frequent updates or significant changes, which could affect the iterative development process and deadline adherence.

Assumptions

- User Adoption: How well users will adjust to the new reservation system in terms of resistance and training needs?
- Scalability: The system's architecture may smoothly develop along with company expansion without significant redesigns or performance problems, provided that the selected technologies and architecture will sufficiently enable scalability as the infrastructure grows because of users and features.
- Security: Whether the put in place security measures, such as data encryption and user authentication, can successfully reduce security threats.
- Resources: All essential members of the team are available to commit the necessary time and energy to the project.
- Technical Viability: The selected technologies can manage the anticipated load and complexity of the project.
- Stakeholder Agreement: All stakeholders (users, administrators, management) agree on project goals, scope, and functions.
- Data Availability: The existing information can be properly transferred or incorporated into the fresh platform with minimum disturbance.

Risks

- Technical Risks: These risks include compatibility concerns, errors in code, performance bottlenecks, restrictions of third-party resources like Supabase (database management challenges), and integration difficulties that can affect functionality and user experience.
- Security Risks: Risks linked with potential weaknesses in the user verification system, as well as data breaches caused by inadequate security measures.
- Integration Risks: Which include compatibility issues with existing systems and services from third parties, API compatibility, and communication methods (e.g. user authentication, reporting tools, real-time availability) that may cause data inconsistencies.

- **User Acceptance Risks:** Risks associated with user comfort and adoption, such as usability concerns, absence of needed functionality, or stakeholder opposition. Users may not find the device intuitive or user-friendly.
- **Scalability Risks:** These risks refer to the system's ability to efficiently grow to meet increased user loads and data capacities without sacrificing performance or functionality.
- **Schedule Delays:** May occur due to resource constraints, technological obstacles, or integration concerns.
- **Budget Overruns:** May occur due to unexpected expenses or inefficiency.
- **Quality Risks:** Defects in the system might result in a negative user experience, booking problems, or security breaches.
- **Testing Problems:** These risks include insufficient test coverage, a shortage of trained testers, and changing requirements, which can hinder complete testing and risk identification.
- **User Adoption:** Being unfamiliar or lack of learning might hinder user acceptance of the new booking system.

Dependencies

- **Using Third-Party Services:** These, like Supabase for database administration and Lucidchart for diagramming might provide risks due to service availability, upgrades, and capability changes.
- **Using Development Technologies:** Such as Visual Studio Code and Git/GitHub for version control can provide challenges for team members in terms of stability, compatibility, and learning.
- **Agile Technique:** Which emphasizes iterative development and frequent releases, can provide challenges for flexibility, stakeholder involvement, and handling changing needs.
- **External Dependencies:** Such as payment gateways, Zoom, and Lucidchart, might provide problems if they are unavailable or disrupted.
- **Internal Dependencies:** Different development teams (front-end and back-end) provide their functionality on schedule and to specification. Timely submission of training materials and data is required to achieve user acceptance.

Testing Priorities

REQUIREMENT/USER STORY	PRIORITY	PROBABILITY	IMPACT	AFFECTS STAKEHOLDERS	RISK
User-friendly interface	High	High	High	Customer, Admin, Accountant	TR, SR, QR, UA
User registration and login	High	High	High	Customer, Admin, Accountant	TR, SR, QR, UA
Booking creation and updating	High	High	High	Customer, Admin,	TR, SR, QR, UA

				Accountant	
Payment processing	High	High	High	Admin, Accountant	TR, SR, QR, UA, BO
Administrative Dashboard functionality	High	High	High	Admin	TR, SR, QR, UA, BO
User Profile Management functionality	High	High	High	Customer, Admin, Accountant	TR, SR, QR, UA, BO
Hotel Search functionality	High	High	High	Customer, Admin	TR, SR, QR, UA
Payment Processing Issues	High	Medium	High	Admin, Developer	TR, QR, UA
Security features	High	High	High	Customer, Admin, Accountant	TR, SR, QR, UA, BO
Create new post	High	High	High	Admin	TR, SR, QR, UA, SD, BO, QR, TP
Mobile responsiveness	Medium	Medium	Medium	Customer, Admin	TR, QR, UA
Feedback and review system	Medium	Medium	Medium	Customer, Admin	TR, QR, UA
Billing and invoicing	Low	Low	Low	Accountant	TR, QR, UA
Advanced search and filtering	Low	Low	Low	Customer, Admin	TR, QR, UA
Hardware Compatibility	NA	NA	NA	NA	NA
Non-functional requirements	NA	NA	NA	NA	NA
Extensive user acceptability testing (UAT)	NA	NA	NA	NA	NA

TR - Technical risks, SR - Security risks, QR - Quality Risks, UA - User Adoption, SD - Schedule delays, BO - Budget overruns, QR - Quality Risks, TP - Testing problems.

Risk owner Roles and Responsibilities

Stakeholders:

- Project Sponsor
- Product Owner
- End Users (Customers)
- Development Team

- Administrative Staff

1. Name: Shweta Udaysinh Role: Project Manager

Responsibilities: The one who monitors the progress of the project and ensures that it is completed on time and within the budget. He may also develop and maintain the project plan such as implementation, work plans, and resources.

2. Name: Ronakkumar Ashokbhai , Kashish Rajeshbhai Role: Systems Analyst

Responsibilities: The one who analyzes the data that could possibly be injected into the system. They may also be responsible for giving technical solutions when problems occur in the project.

3. Name: Remya Shaji Role: Researcher

Responsibilities: The one responsible for finding relevant and accurate topics that are related to the system.

Test Levels

Unit Testing

1.1 Room Availability:

- Test Cases:
 - o After booking, confirm that the accommodation is indeed shown as unavailable.
 - o Confirm that you receive an error when you try to reserve a room that is already reserved.
 - o Make sure the system shows the availability of rooms for various dates.

2.1 Guest Information:

- Test Cases:
 - o Verify that the guest's details, including name, email address, and phone number, are input accurately.
 - o Examine how errors are handled when data formats are invalid (e.g., email without the "@" symbol).
 - o Check to see if visitor data is correctly saved and retrieved to confirm data persistence.

3.1 Booking Process:

- Test Cases:
 - o Test through successful reservation scenarios using various room categories and lengths of stay.
 - o Verify that the cancellation process for a booking is successful and that the room availability is updated.
 - o Check that pricing calculations considered the kind of room, the length of stay, and any relevant discounts.

4.1 Payment Processing:

- Test Cases:
 - o Use various payment mechanisms to simulate successful and unsuccessful payment transactions (e.g., credit card, debit card).

- o Assure appropriate error handling in the event of incomplete or inaccurate payment information.
- o Make that the customer is not charged by the system if the booking fails.

5.1 User Authentication:

- Test Cases:
 - o Check that you were able to log in using your real credentials.
 - o Test erroneous login attempts by using fictitious passwords or users.
 - o Make careful to handle and store passwords securely (don't save them in plain text).

Integration Testing

2.1 User Management and Booking Integration:

- Test Cases:
 - o Objective: Verify that the booking system and user registration and authentication are integrated.
 - o Test Cases: Try creating reservations for logged-in users. Check that the user's information is accurately linked to the reservations.

3.1 Booking and Room Management Integration:

- Test Cases:
 - o Objective: Verify the proper integration between the room inventory management and the booking management module.
 - o Test Cases: Check if the room availability has changed following a successful reservation. Verify that the booking and room details are in sync.

4.1 Payment Processing Integration:

- Test Cases:
 - o Objective: Verify that the booking system and the payment processing systems work together seamlessly.
 - o Test Cases: Explore with successful and unsuccessful payments. Make sure the system adjusts the booking statuses appropriately.

5.1 Notification System Integration:

- Test Cases:
 - o Objective: Make sure that user management and booking functions with the notification system.
 - o Test Cases: Test the creation and delivery of email confirmations. Make sure users are informed when there are changes or cancellations.

6.1 Database Integration Testing:

- Test Cases:

- o Objective: Verify how the system and database are communicating with one other.
- o Test Cases: Verify that during various processes, data is accurately saved, updated, and retrieved from the database.

System Testing

3.1 Functionality Testing:

- Booking Process:
 - o To make sure consumers can properly search for hotels, choose rooms, make reservations, and receive confirmation, test the entire booking process.
 - o Check how the system responds to various situations, such as unavailable rooms or conflicting reservations.
- User Management:
 - o Check the functionality of user registration, login, and logout.
 - o Check to make sure authentication functions as it should and that user accounts are established securely.
- Hotel Information Management:
 - o Make that the system reflects updates and that hotel details are displayed appropriately.
 - o Evaluate how the system reacts to inaccurate or incomplete hotel information.

4.1 Performance Testing:

- Scalability:
 - o Evaluate the scalability of the system by modelling many concurrent customers making reservations.
 - o Check to see if the system can withstand high loads without experiencing any performance issues.
- Response Time:
 - o Analyse and make sure that crucial procedures, including booking hotels and making bookings, have reaction times that are within reasonable bounds.

5.1 Usability Testing:

- Consider how easy and straightforward the user interface is to use.
- To guarantee a satisfying user experience, test the user interface and navigation.

6.1 Reliability and Availability Testing:

- Examine the system's error-handling and recovery methods to gauge its dependability.
- Examine the system's error-handling and recovery methods to gauge its dependability.

7.1 Compatibility Testing:

- Check that the hotel reservation system functions properly across a range of operating systems, devices, and browsers.
- Verify compatibility across a range of screen dimensions and resolutions.

Performance Testing

4.1 Load Testing:

- Objective: Analyse how well the system performs at peak and anticipated loads.
- Actions:
 - o Create a realistically high number of concurrent booking users.
 - o Determine the system's breaking point by progressively increasing the load.
 - o Analyse system behaviour and response times under various load levels.

4.2 Stress Testing:

- Objective: Evaluate the stability of the system and ascertain its maximum capacity.
- Actions:
 - o To find failure sites in a system, apply loads that are higher than its intended capability.
 - o Keep an eye on how the system behaves under pressure and look for any possible bottlenecks.

4.3 Scalability Testing:

- Objective: Analyse the system's ability to scale as the load increases.
- Actions:
 - o Examine the system's performance with different user counts at the same time.
 - o Check to see if adding more servers or other resources improves performance.

4.4 Response Time Testing:

- Objective: Calculate how long it takes for important operations to react.
- Actions:
 - o Analyse how quickly users can search, make bookings, and cancel them—all common user actions.
 - o Establish reasonable limits for response times and make sure the system reaches them.

4.5 Concurrency Testing:

- Objective: Evaluate the system's ability to manage several users at once.
- Actions:
 - o Test simultaneous use of payment, booking, and other essential features.
 - o Determine and fix any concurrency problems or racial circumstances.

Security Testing

5.1 Data Encryption:

- Objective: Make sure that sensitive data is encrypted both during transmission and storage, including payment information and user credentials.
- Actions:
 - o Make sure that every communication is using HTTPS.
 - o Verify whether robust encryption mechanisms are being used to store data.

5.2 Authentication Testing:

- Objective: Verify that the authentication systems are functioning as intended.
- Actions:
 - o Verify criteria for complexity and test for lax password regulations.

- o Verify whether your account can be locked out following a predetermined number of unsuccessful login attempts.

5.3 Authorization Testing:

- Objective: Verify that users possess the proper authorizations and access rights.
- Actions:
 - o Examine user permissions and responsibilities for different functionalities (e.g., admin, regular user).
 - o Verify that unauthorized people are unable to enter places that are restricted.

5.4 Input Validation:

- Objective: Check sure user inputs are validated and sanitized by the system to avoid common vulnerabilities.
- Actions:
 - o Check for injection threats such as cross-site scripting (XSS) and SQL injection.
 - o Input field validation is necessary to stop data manipulation.

5.5 Error Handling:

- Objective: Make sure that private information is not disclosed in error messages.
- Actions:
 - o Check that error messages are generic and do not reveal system information by testing them.
 - o Create personalized error pages to provide a safer online experience.

Regression Testing

6.1 Test Planning:

- Objective: Arrange regression testing in accordance with the system modifications.
- Actions:
 - o Determine which parts of the hotel reservation system have been updated or had problem patches applied.
 - o Ascertain which essential features the regression test suite ought to have.

6.2 Automate Test Cases:

- Objective: Regression testing can be streamlined by automating important and repetitive test cases.
- Actions:
 - o Use testing frameworks to automate test scenarios, such as JUnit and Selenium.
 - o Give priority to automated tests for essential and often used features.

6.3 Data Integrity Testing:

- Objective: Check to see that data integrity is preserved both before and after modifications.
- Actions:
 - o Examine how updates affect the system's current data.
 - o Verify that data consistency is maintained throughout different components.

6.4 User Interface Testing:

- Objective: Verify the consistency of the UI elements and interactions.
- Actions:

- o Look for any UI adjustments brought about by recent updates.
- o Make sure users have the same level of ease navigating the system.

6.5 Cross-Browser and Cross-Device Testing:

- Objective: Assure consistent functionality between various devices and browsers.
- Actions:
 - o Verify system compatibility by testing it across a range of devices and browsers.
 - o Verify that the most recent modifications had no effect on the UI's responsiveness.

User Acceptance Testing (UAT)

- Objective: By verifying the functioning and usability of the hotel booking system, UAT makes sure it satisfies end users' expectations and requirements.
- Scope: The main goal of UAT is to test common user tasks like booking reservations, finding hotels, and managing bookings.
- Test Scenarios:
 - o Looking for hotels according to interests, dates, and location.
 - o Looking at hotel data, available rooms, and cost details.
 - o Choosing a room type, a date for the reservation, and inputting guest information are all part of making a reservation.
 - o Taking care of reservations, including changing or cancelling them.
 - o Accessing user profiles, checking booking history, and accessing account information.
- Execution: Testers carry out predetermined test scenarios and record any problems they run into. These testers can be ending users or specially assigned testers.
- Validation Criteria: A user-friendly interface, effective task completion, precise information, seamless interactions, and data security are examples of success criteria.
- Feedback and Iteration: Enhancing the user experience and addressing concerns are guided by feedback from users.
- Sign-off: A formal approval signifies that the product is ready to be used and deployed in a production setting.

Environmental Requirements

1. Hardware Requirements:
 - Server Infrastructure: Robust server hardware could be needed for the system to effectively manage database transactions and concurrent user requests.
 - Network Infrastructure: Reliable and fast network connectivity is necessary to ensure smooth communication between client devices, servers, and databases.
2. Software Requirements:
 - Operating System: Indicate the operating system or systems the server and client components are compatible with. Popular options include cloud-based services, Linux, and Windows Server.
 - Web Server: Determine which web server software—such as Apache or Nginx—is needed to run the hotel reservation system.
 - Database Management System: Indicate the supported database management system (DBMS) for data storing and retrieval (e.g., MySQL, PostgreSQL, MongoDB).

3. The Stack of Technologies:

- Programming Languages: Identify the programming languages (such as Java, Python, and JavaScript) that are utilized in system development.
- State any frameworks or libraries (such Django, React, Spring Boot, etc.) that are used for system development.

4. Scalability Requirements:

- Plan for Scalability: Describe how the system can grow to accommodate more users. Think about cloud scalability alternatives, distributed design, and load balancing.

5. Security Requirements:

- Encryption: Apply encryption technologies to protect data transfer between clients and servers, such as SSL/TLS.
- Authorization & Authentication: Specify user authorization and authentication procedures to guarantee safe system access.
- Systems for Intrusion Detection & Firewalls: Utilize security measures to guard against any threats and unlawful access.

6. Data Backup and Recovery:

- Backup Plan: Establish an effective information backup plan to guard against losing data in the event of a system breakdown.
- Procedures for Recovery: Establish protocols for data recovery and system restoration to operation following an incident.

7. Compliance Requirements:

- Laws for Safeguarding Data: Make that the system complies with all applicable privacy and data protection regulations (such as the CCPA and GDPR) in the areas where it operates.
- Industry-Specific Regulations: Follow particular to industry guidelines and rules that are relevant to the lodging and hospitality industries.

8. Performance Requirements:

- Response Time: To guarantee a satisfying user experience, specify acceptable response times for user interactions.
- System Throughput: Specify how many bookings or transactions the system can process in a certain amount of time.

9. Aspects of the Environment:

- Conditions in the Server Room: Ensure that server rooms have the best possible environmental conditions, including proper ventilation and temperature control.
- Plans for Disaster Recovery: Create plans for disaster recovery that consider unforeseen circumstances such as power outages, natural disasters, and other situations.

10. Accessibility for Users:

- **Compatibility across Browsers:** Make sure it works with most web browsers to give users a consistent experience.
- **The Responsiveness of Mobile:** Create the system with accessibility and usability in mind for a range of devices, such as tablets, smartphones, and desktop computers.

Acceptance Criteria

Functional Acceptance Criteria

A hotel booking system's functional acceptance criteria ought to outline the criteria that the system must achieve to be regarded operationally successful. These standards should be determined by the system's primary functionality and user experience, making sure the system for bookings is effective, precise, and customer friendly. Following are some important functional acceptability criteria for a hotel reservation system:

- **Booking Establishment:** Clients are supposed to be allowed to make a booking by choosing a room type, check-in and out dates, and the number of guests. The program ought to estimate the overall cost depending on the room type and period of stay.
- **Room Availability:** The system must appropriately show the availability of rooms according to the dates and room type chosen. Clients are likely unable to reserve rooms which have been reserved for the provided dates.
- **Reservation Modification:** Clients should have the ability to alter their booking information, including check-in/check-out times, room type, or the number of people staying there, within the period provided.
- **Reservation Cancellation:** Client's ought to be permitted to terminate the reservation during the cancellation window. The system ought to process any appropriate reimbursements in accordance with the hotel's cancellation policies.
- **Customer Profile Management:** Clients should be accessible to establish, handle, and gain access to their accounts. This entails setting up a profile, signing in, and modifying private data.
- **System Reliability and Availability:** Clients should have been capable of utilizing the booking system with little downtime.

Performance Acceptance Criteria

Performance acceptance requirements for a hotel reservation system usually involve verifying that the system can manage a given amount of volume and stress while preserving appropriate time to respond and reliability.

- **Response Time:** The platform must respond to user activities (for instance, requesting availability, making a reservation, or withdrawing a reservation) during a certain time frame, including a couple of seconds for most client-facing activities and a maximum of five seconds for more complicated procedures.
- **Throughput:** The platform should be capable to cope with a specified number of transactions per second (for example, searches, reservations, and transactions) without compromising performance, ensuring good availability regardless of high-usage periods.
- **Scalability:** The system ought to expand seamlessly with rising demand, allowing for an increasing number of simultaneous users and operations while maintaining performance.
- **Availability:** The system must have an excellent availability rate, with an uptime of a minimum of 99.9%, to guarantee clients have consistent access at all instances.
- **Error Rate:** The system is supposed to have a low rate of mistakes (less than 0.1% of operations leading to failures) and manage errors graciously.

- Data Consistency: The platform must render data consistent throughout every module and actions, including guaranteeing that booking data are precisely posted in real time.

Acceptance Test

Purpose:

Acceptance testing serves the objective of verifying that a hotel booking system satisfies company standards and provides end customers with the anticipated degree of functionality and performance. It is the last stage of testing preceding the system goes into production, and it is a crucial validation phase that guarantees the system is prepared for real-world use.

Acceptance Test Plan:

Scope:

The acceptance test approach to the hotel reservation system comprises comprehensive testing to make sure that the system satisfies company standards and delivers the best possible user experience. This includes evaluating the full booking experience, from finding available rooms to making a reservation and obtaining confirmation. This additionally entails evaluating the cancellation process, client management of accounts (for example, establishing an account, login, and profile updates). The strategy also includes testing to ensure system accessibility, accuracy of data, and compatibility with third-party resources (such as payment gateways and email notifications). The purpose is to test the system's capabilities, efficiency, and dependability in real-world circumstances.

Out of Scope:

The acceptance test plan eliminates individual tests and specialized module-level testing, which are being addressed via different testing strategies. Furthermore, while certain functionality and load testing may be included during acceptance testing, significant performance testing is usually done independently. The strategy notably does not evaluate the technical specifics of services provided by third parties, instead focusing on their general interface with the system. Furthermore, non-functional testing, such as compliance and safety, can be addressed independently in separate test plans.

Environment Used:

The software system will be developed and tested in an environment conducive to web development, likely including web servers, databases, and development frameworks such as HTML, CSS, Selenium and server-side languages like PHP, Python. Testing will be conducted in a controlled environment to ensure accurate assessment of functionality.

Server Environment:

- a. Operating System: Windows 11
- b. Front-End Tools: ReactJs
- c. Back-End Tools: Postgres

Test Data:

1. The test data for the Hotel Booking Management System project covered a wide range of scenarios, including user registration, login, password management, business and customer details management, order processing, and search functionality.
2. It incorporated diverse user roles such as admin, customer, and business, ensuring comprehensive testing of all system functionalities.
3. The test data was meticulously designed to simulate real-world usage scenarios, guaranteeing that the system's behavior was thoroughly evaluated under various conditions.

Testing Scenarios:

1. The testing scenarios for the Hotel Booking Management System project were meticulously designed to cover all aspects of the system's functionality outlined in the project scope.
2. Positive and negative test cases were incorporated into the scenarios to assess the system's behavior under different conditions, including valid and invalid inputs, error handling, and boundary conditions.
3. The testing approach ensured a comprehensive evaluation of the system's performance and reliability, instilling confidence in its ability to meet user requirements and expectations.
4. Through rigorous testing, the system's response to various scenarios was thoroughly assessed, contributing to its readiness for deployment.
5. Continuous monitoring and periodic updates should be implemented to address any potential vulnerabilities or performance issues that may arise in future versions of the system

Tools Used:

1. Automated testing tools were employed to streamline the testing process and ensure efficiency.
2. Tools such as Selenium WebDriver were utilized for automated testing of web application functionalities.

Automated testing tools helped in executing repetitive test cases and validating the consistency and reliability of the " Hotel Booking Management System " software system across different environments and scenarios.

This comprehensive testing environment facilitated rigorous evaluation of the " Hotel Booking Management System " software system, ensuring its functionality, compatibility, and reliability across various operating systems, web browsers and user scenarios.

Acceptance Test Cases:

This document describes the acceptance test cases used to ensure the operation of the hotel booking system. These tests will ensure that the system allows users to search for hotels, choose preferred room types and dates, appropriately compute price, securely process guest information, and provide successfully confirmations for reservations that meet all user needs.

Business Acceptance Test Cases:

Business Acceptance Test Case Details:

Test Case 1: Booking Room Successfully

Test Case ID: FR5_BA_TC001

Title: Booking Room Successfully

Description: This test confirms that a user may successfully reserve a hotel using accurate data.

Steps:

- Go to the hotel reservation system and sign in as a customer who previously booked.
- Browse for hotel rooms by choosing your preferred arrival and departure dates.
- Choose a room category from the listing of available rooms.

- Enter the guest's name, email, and phone number.
- Provide accurate payment information and finish the payment process.
- Check that the reservation confirmation page has the booking details.

Actual: The user successfully reserved the room and reviewed the booking information on the booking confirmation page.

Expected: The user must successfully reserve the room and review the booking details on the confirmation page.

Test Case 2: Search for Available Rooms

Test Case ID: FR4_BA_TC002

Title: Search for Available Rooms

Description: This test ensures that the application precisely shows available rooms within the requested date range and criteria.

Steps:

- Visit the room search website.
- Enter your check-in and out dates.
- Choose your preferred filters, including room type, facilities, or vacancy.
- Hit the "Search" button.
- Check that your search results show accommodations available for the selected time frame and criteria.

Actual: The results of the search showed a listing of available rooms that met the specified criteria.

Expected: The results of a search should include a listing of available rooms that meet the specified parameters.

Test Case 3: Reservation Cancellation

Test Case ID: FR10_BA_TC003

Title: Reservation Cancellation

Description: This test ensures that a user can effectively terminate the reservation and receive a confirmation.

Steps:

- Sign in as a current client.
- Proceed to the Booking History or My Reservations page.
- Choose a reservation to cancel.
- Confirm the cancelation request.
- Check that the booking has been deleted and the message of confirmation has been displayed.

Actual: The reservation was abandoned, and the user receives a cancellation confirmation message.

Expected: The booking must be cancelled, and the person making the reservation ought to get a cancellation confirmation message.

Test Case 4: Guest Account Creation

Test Case ID: FR2_BA_TC004

Title: Guest Account Creation

Description: This test demonstrates that a user can establish a new guest account.

Steps:

- Proceed to the registration form.
- Provide accurate details such as your first and last name, email, and password.

Actual: The guest succeeded in creating an account.

Expected: The guest should have successfully created an account.

User Acceptance Test Cases:

Test Case 1: Registration

Test Case ID: FR1_UA_TC001

Test Case Description: Verify that a new user can register on the system.

Steps:

- Navigate to the registration page.
- Enter valid registration details (e.g., name, email, password).
- Click on the "Register" button.

Actual: User successfully registered.

Expected Result: User should be successfully registered and redirected to the login page.

Test case 2: Login

Test Case ID: FR2_UA_TC001

Test Case Description: Verify that a registered user can log in to the system.

Steps:

- Navigate to the login page.
- Enter valid credentials (email and password).
- Click on the "Login" button.

Actual: The guest succeeded in login.

Expected Result: User should be successfully logged in and redirected to the dashboard.

Test Case 3: Admin Room Management

Test Case ID: FR8_UA_TC001

Test Case Description: Verify that an admin can manage rooms (add, edit, delete).

Steps:

- Navigate to the room management section.
- Add a new room by providing necessary details (e.g., room type, price).
- Edit an existing room's details (e.g., price, availability).
- Delete a room from the system.

Actual: The guest succeeded in creating an account.

Expected Result: Admin should be able to add, edit, and delete rooms successfully.

Test Case 4: User Booking Management

Test Case ID: FR5_UA_TC001

Test Case Description: Verify that a user can manage their bookings (view, modify, cancel).

Steps:

- Navigate to the booking management section.
- View the list of existing bookings.
- Modify the details of a booking (e.g., dates, room type).
- Cancel a booking.

Actual: The guest succeeded in booking a room.

Expected Result: User should be able to view, modify, and cancel their bookings successfully.

Test Case 5: User Searching Management

Test Case ID: FR4_UA_TC001

Test Case Description: Verify that users can search for available rooms.

Steps:

- Enter search criteria (e.g., check-in/out dates, room type).
- Click on the "Search" button.

Expected Result: System should display a list of available rooms based on the search criteria.

Test Case 6: Admin Create New Post

Test Case ID: FR3_UA_TC001

Description: Admin successfully creates a new post.

Pre-Conditions: Admin user is logged in.

Steps:

- Navigate to the page.
- Enter a valid title for the post.
- Enter the post content in the designated field.
- Select the appropriate category for the post.
- Click on button.

Expected Result: The new post is created successfully and appears on the website/platform.

Actual: Pass the post is created and displayed as expected

Test Case ID: FR3_UA_TC002

Description: Admin tries to create a post with an empty title.

Pre-Conditions: Admin user is logged in.

Steps:

- Navigate to the page.
- Leave the title field blank.
- Enter valid content in the body field.
- Click on button.

Expected Result: An error message appears indicating that a title is required.

Actual: Pass an error message for missing title is displayed.

Test Case ID: FR3_UA_TC003

Description: Admin tries to create a post with invalid content.

Pre-Conditions: Admin user is logged in.

Steps:

- Navigate to the page.
- Enter a valid title for the post.
- Enter gibberish or excessive formatting in the body field.
- Click on button.

Expected Result: An error message appears indicating that the content is invalid.

Actual: Pass an error message for invalid content is displayed.

Test case 7: User Cancellation Booking

Test Case ID: FR7_UA_TC005

Description: User successfully cancels their existing booking.

Pre-Conditions: User is logged in and has an existing booking.

Steps:

- Navigate to the page.
- Select the booking they want to cancel.
- Click on the button.
- Confirm the cancellation if prompted.

Expected Result: The booking is cancelled successfully, and the user receives. Their booking list should no longer show the cancelled booking.

Actual: Pass the booking is cancelled and reflected in the user's account

Test case 8: Admin Cancellation Booking

Test Case ID: FR10_UA_TC007

Description: Admin successfully cancels a user's booking.

Pre-Conditions: Admin user is logged in and can access user bookings.

Steps:

- Navigate to the page.
- Locate the user's booking that needs cancellation.
- Click on the button associated with that booking.
- Confirm the cancellation.

Expected Result: The booking is cancelled successfully, and the user receives notification. The booking status should reflect the cancellation in the admin panel.

Actual: Pass the booking is cancelled and reflected in both admin and user accounts.

Additional Test Cases:

Compatibility Testing:

Browser Compatibility:

Title: Browser Compatibility

Description: This test verifies that the hotel booking system functions correctly across different web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

Steps:

- Access the web application using Google Chrome.
- Perform various actions and functionalities.
- Repeat steps 1-2 for Mozilla Firefox, Safari, and Microsoft Edge.

Actual: The system functions properly across all tested web browsers.

Expected: Consistent performance and functionality across different web browsers.

Performance Testing:

Load Testing:

- Title: Load Testing
- Description: Load testing evaluates the system's performance under heavy user traffic to ensure it can handle concurrent interactions without slowdowns or crashes, providing insights into its scalability and robustness.
- Steps:

Gradually increase the number of concurrent users accessing the application.

Monitor system response times and resource consumption metrics.

Assess the system's performance and stability under peak load conditions.

- Actual: The system demonstrated consistent and acceptable performance metrics even when subjected to heavy loads, indicating its ability to scale effectively.
- Expected: The system is expected to maintain optimal performance levels and handle increased user traffic without experiencing degradation or disruptions, ensuring a seamless user experience even during peak usage periods.

Stress Testing:

- Title: Stress Testing
- Description: Applies extreme loads to the system beyond its normal capacity to identify any breaking points or performance degradation, ensuring robustness and reliability under stress.
- Steps:
- Introduce extreme levels of concurrent user traffic to the application.
- Monitor system behaviour and response times.
- Assess system stability and identify any performance degradation or failure points.
- Actual: System remains stable and responsive under stress.
- Expected: Application withstands extreme loads without crashing or experiencing significant performance degradation.

Scalability Testing:

- Title: Scalability Testing
- Description: Evaluates the system's ability to scale up or down dynamically based on changing user demands, ensuring smooth operation during periods of high traffic as well as resource optimization during low usage.
- Steps:
- Simulate varying levels of user traffic on the application.
- Monitor system response times and resource utilization.
- Assess the system's ability to scale up and down dynamically.
- Actual: System scales effectively based on changing user demands.
- Expected: Application dynamically adjusts resources to maintain performance and optimize resource utilization.

Security Testing:

Authentication Testing:

- Title: Authentication Testing
- Description: Verifies the strength and effectiveness of the authentication mechanisms, including password encryption, session management, and multi-factor authentication, to prevent unauthorized access.
- Steps:
- Test password encryption and hashing algorithms.
- Review session management mechanisms.
- Verify the implementation of multi-factor authentication where applicable.
- Actual: Authentication mechanisms are robust and secure.

- Expected: Unauthorized access is prevented through effective authentication mechanisms.

Authorization Testing:

- Title: Authorization Testing
- Description: Tests the system's authorization controls to ensure that users can only access functionalities and data appropriate to their roles, preventing unauthorized actions.
- Steps:
 - Test access control for different user roles.
 - Attempt to access restricted functionalities or data.
 - Verify that unauthorized access attempts are blocked.
 - Actual: Authorization controls restrict access appropriately.
 - Expected: Users can only access functionalities and data relevant to their roles.

Data Protection Testing:

- Title: Data Protection Testing
- Description: Checks for vulnerabilities such as SQL injection, cross-site scripting (XSS), and data leakage, ensuring the confidentiality, integrity, and availability of sensitive information stored and transmitted by the system.
- Steps:
 - Perform security scans for common vulnerabilities.
 - Test for SQL injection and XSS attacks.
 - Review data encryption mechanisms.
 - Verify that sensitive data is adequately protected.
- Actual: System is secure against common data vulnerabilities.
- Expected: Sensitive data is protected from unauthorized access and manipulation.

These test cases cover a wide range of functionalities and scenarios to thoroughly evaluate the " Hotel Booking Management System " software system and ensure its readiness for deployment.

Acceptance Test Case Results:

In this section, the results of executing all acceptance test cases for the " Hotel Booking Management System project " software system are presented. The test statistics, including tests run, tests passed, tests failed, and any relevant comments are displayed in a clear and concise tabular format.

Test Case	Tests Run	Tests Passed	Tests Failed	Comments
Business Acceptance	4	4	0	All business-related functionalities passed without issues.
User Acceptance	10	10	0	All user-related functionalities passed successfully.
Search Functionality	3	3	0	Search capability was successfully tested, yielding accurate and relevant results for users.
Compatibility Testing	2	3	0	Application compatibility was evaluated across browsers, operating systems, and mobile devices, and no concerns were discovered.
Performance Testing	3	3	0	System performance tested under various load conditions, demonstrating satisfactory scalability and responsiveness.
Security Testing	3	3	0	Application security tested against common vulnerabilities, ensuring robust protection of user data and system integrity.

Conclusion

The Hotel Booking Management System project has successfully passed acceptance testing, showcasing its robust performance, reliability, and security across all tested functionalities and scenarios. With no failed test cases, the system proves its adherence to the specified requirements and readiness for deployment. However, ongoing monitoring and updates are recommended to promptly address any emerging vulnerabilities or performance concerns in subsequent iterations.

System Test

Purpose

The purpose of this document is to present the outcomes of the system testing conducted for the Hotel Booking Management System. It serves to communicate the findings, successes, and areas for improvement identified during the testing process. This document aims to provide stakeholders, including project sponsors, developers, testers, and end-users, with a comprehensive overview of the system's performance and reliability. By documenting the test results, this report facilitates informed decision-making regarding the system's readiness for deployment. Furthermore, it offers insights into any potential risks, issues, or areas requiring further attention, thereby guiding future development efforts and ensuring the delivery of a high-quality product aligned with project objectives.

System Test Plan

Scope

2.1.1 Manual System Testing: Verify all aspects of the Hotel Booking Management System manually, including user interfaces, functionalities, and workflows.

2.1.2 Automated System Testing: Utilize automated testing tools and frameworks to validate the system's functionalities, including user authentication and booking processes.

2.1.3 Database System Testing: Verify the integrity, accuracy, and performance of the database system used in the Hotel Booking Management System, including data storage, retrieval, and manipulation operations.

2.1.4 Performance System Testing: Evaluate the system's performance under various load conditions, ensuring scalability, responsiveness, and efficiency during peak usage periods.

2.1.5 Security System Testing: Assess the system's security mechanisms, including data encryption, user authentication, authorization, input validation and error handling.

2.1.6 Usability System Testing: Evaluate the user-friendliness and ease of use of the Hotel Booking Management System, focusing on factors such as navigation, intuitiveness, clarity of instructions, and overall user experience.

2.1.7 Compatibility System Testing: Assess the compatibility of the Hotel Booking Management System across different platforms, devices, browsers, and operating systems, ensuring seamless functionality and consistent user experience across various environments.

2.1.8 Functional System Testing: Verify the functional requirements of the Hotel Booking Management System, ensuring that each feature and functionality operates according to specifications and meets the intended objectives of the system.

Out of Scope

2.2.1 Interoperability Testing: Testing the integration of the Hotel Booking Management System with other systems or databases, including third-party services or APIs, is not within the scope of system testing.

2.2.2 Business Continuity and Disaster Recovery Testing: Evaluating the system's resilience to errors or calamities, as well as testing for business continuity and disaster recovery procedures, is not included in system testing.

2.2.3 Compliance Testing: Ensuring compliance with industry-specific regulations such as HIPAA or GDPR, as well as testing for business continuity and disaster recovery procedures, falls outside the scope of system testing.

2.2.4 Compatibility Testing with Specific Devices or Browsers: While general compatibility testing is included in the scope, testing for compatibility with specific devices, browsers, or screen resolutions may not be covered under system testing.

2.2.5 Environmental Requirements: Testing environmental factors such as hardware, software, scalability, security, and performance requirements is not part of system testing.

Environment Used

Development Environment: Supabase for database, ReactJs.

Testing Environment: For performance Testing – Jmeter, for security testing – Burp Suite, for database testing - .

Production Environment: Windows 11, Visual Studio Code.

System Test Cases

2.1.1 Manual System Testing:

Verify all aspects of the Hotel Booking Management System manually, including user interfaces, functionalities, and workflows.

FR1_MS_TC001: Verify User Registration Process

Steps:

- Access the user registration page.
- Enter valid user details (e.g., name, email, password).
- Submit the registration form.

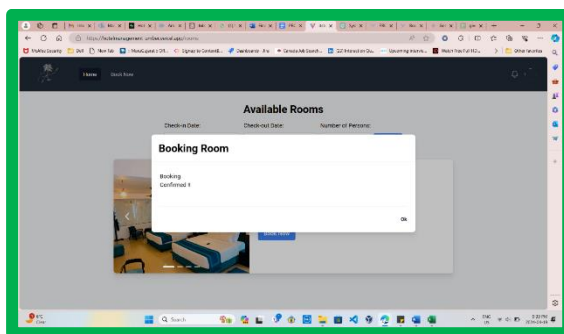


Expected Result: User should be successfully registered, and a confirmation message should be displayed.

FR5_MS_TC002: Verify Room Booking Process

Steps:

- Login with valid user credentials.
- Navigate to the room booking section.
- Select desired room type and dates.
- Confirm the booking.

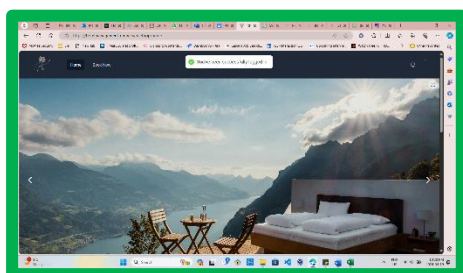


Expected Result: Booking should be successfully processed, and the user should receive a confirmation email.

FR2_MS_TC003: Verify Admin Login Functionality

Steps:

- Open the admin login page.
- Enter valid admin credentials.
- Click on the login button.

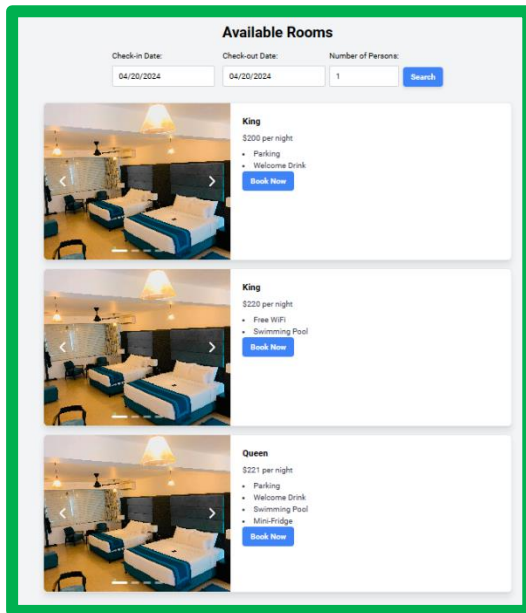


Expected Result: Admin should be logged in successfully, and the admin dashboard page should be displayed.

FR4_MS_TC004: Verify Room Search by Date

Steps:

- Navigate to the room search section.
- Enter a specific Date in the search filter. (20-april-2024 to 20-april-2024)
- Initiate the search.



Expected Result: System should display available rooms for the specified date, matching the search criteria.

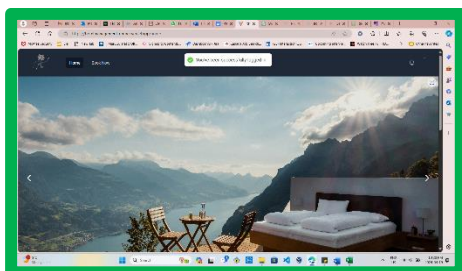
2.1.2 Automated System Testing:

Utilize automated testing tools and frameworks to validate the system's functionalities, including user authentication and booking processes.

FR2_AS_TC001: Verify User Login Functionality

Steps:

- Open the login page.
- Enter valid user credentials.
- Click on the login button.



Expected Result: User should be logged in successfully, and the dashboard page should be displayed.

FR4_AS_TC002: Verify Room Availability Check

Steps:

- Navigate to the room search section.
- Enter desired dates and room preferences.
- Initiate the availability check.

The image shows two screenshots of a 'Booking Room' form. The left screenshot shows the form with a green border and a blue highlight on the 'Check-out Date' field. The right screenshot shows the form with a green border and a blue highlight on the 'Check-in Date' field. Both screenshots show the form fields for First name, Last name, Email, Date of Birth, Phone Number, Address, Number of Adults, Number of Children, Add Ons, Extra Services, and Price. The form is titled 'Booking Room' and has a 'Cancel' button at the bottom right.

Expected Result: System should display available rooms matching the criteria specified by the user.

2.1.3 Database System Testing:

Verify the integrity, accuracy, and performance of the database system used in the Hotel Booking Management System, including data storage, retrieval, and manipulation operations.

DT_TC001: Verify Data Retrieval Operation

Steps:

- Access the user profile section.
- Retrieve user profile data from the database after clicking on book now option the user can see all the registered details on the screen which are necessary for the room booking.

The image shows a screenshot of a 'Booking Room' form with a green border. The form displays the user's profile data, including First name, Last name, Email, Date of Birth, Phone Number, Address, Number of Adults, Number of Children, Add Ons, Extra Services, and Price. The form is titled 'Booking Room' and has a 'Cancel' button at the bottom right.

Expected Result: User profile data should be retrieved accurately from the database and displayed on the user interface.

DT_TC001: Verify Data Integrity Check

Steps:

- Input test data into the system.
- Retrieve data from the database.
- Compare input and retrieved data.

Booking Room

First name: Last name:

Email: Date of Birth:

Phone Number: Address:

Number of Adults: Number of Children:

Add Ons: Extra Services:

Price:
Room: 400 for 2 days
Room Add Ons: 0 for 2 days
Extra Add Ons: 0 for 2 days
Total Price: 400

Expected Result: Input and retrieved data match, indicating data integrity.

2.1.4 Performance System Testing:

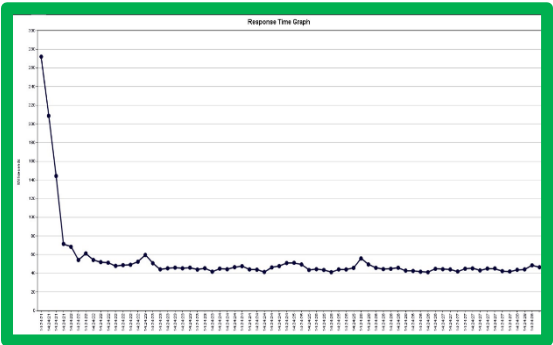
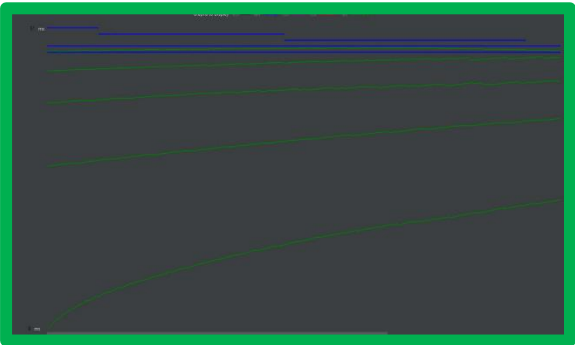
Evaluate the system's performance under various load conditions, ensuring scalability, responsiveness, and efficiency during peak usage periods.

NFR1_PS_TC001: Verify System Responsiveness Under Load

Steps:

- Simulate a high volume of concurrent user requests. (10,000 users)
- Monitor system response times for various operations.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	10000	46	0	366	16.98	0.00%	1467.8/sec	2062.85	249.41	1439.1
TOTAL	10000	46	0	366	16.98	0.00%	1467.8/sec	2062.85	249.41	1439.1

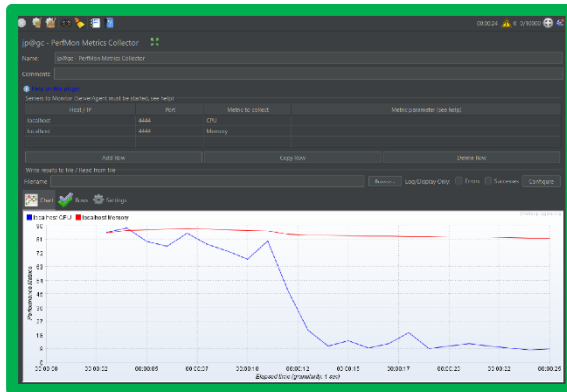


Expected Result: System should remain responsive and maintain acceptable response times even under peak load conditions.

NFR1_PS_TC002: Verify System Scalability

Steps:

- Gradually increase the load on the system by adding additional virtual users.
- Monitor system performance metrics such as CPU utilization and memory usage.



Expected Result: System should scale effectively to handle increased load without significant degradation in performance.

2.1.5 Security System Testing:

Assess the system's security mechanisms, including data encryption, user authentication, authorization, input validation and error handling.

NFR4_SS_TC001: Verify Input Validation for User Registration

Steps:

- Attempt to register a user with invalid input data (e.g., special characters in the name field).

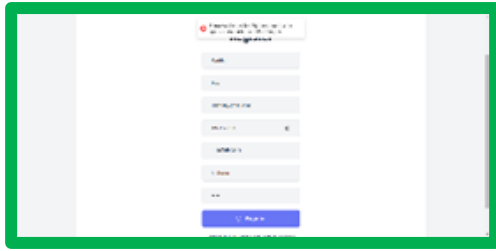


Expected Result: System should reject the registration attempt and display appropriate error messages for invalid input.

NFR4_SS_TC002: Verify Password Encryption

Steps:

- Navigate the registration page.
- Try to enter password and check its visibility.



Expected Result: Password should be securely encrypted and stored in the database, ensuring protection against unauthorized access.

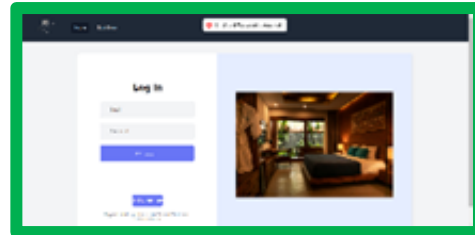
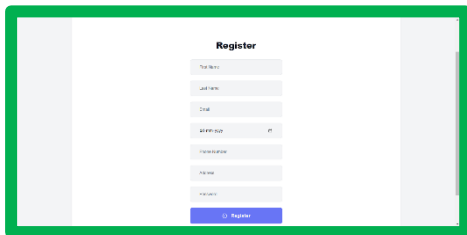
2.1.6 Usability System Testing:

Evaluate the user-friendliness and ease of use of the Hotel Booking Management System, focusing on factors such as navigation, intuitiveness, clarity of instructions, and overall user experience.

NFR2_US_TC001: Verify Navigation Ease

Steps:

- Perform common user tasks such as registration, login, and room booking.
- Assess the intuitiveness of navigation and ease of locating functionalities.

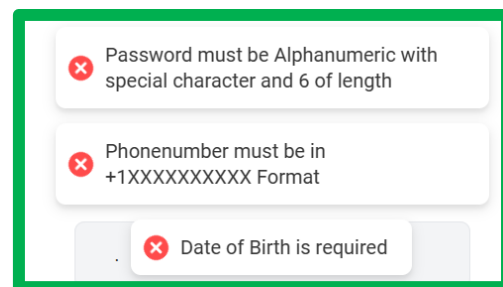
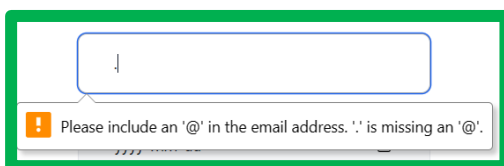


Expected Result: Users should find it easy to navigate through different sections of the application and perform tasks without confusion.

NFR2_US_TC002: Verify Clarity of Instructions

Steps:

- Access various sections of the application.
- Review the instructions provided for different tasks.



Expected Result: Instructions should be clear and concise, guiding users effectively on how to perform various actions within the application.

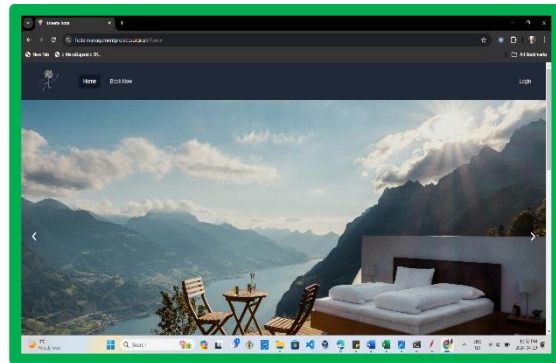
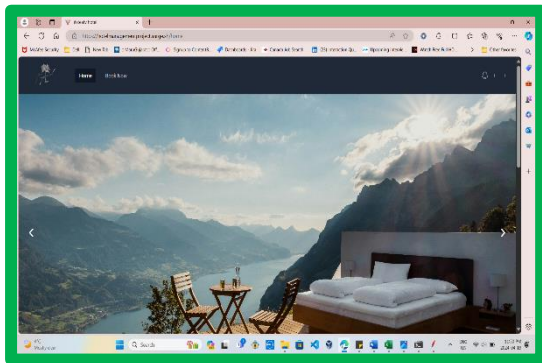
2.1.7 Compatibility System Testing:

Assess the compatibility of the Hotel Booking Management System across different platforms, devices, browsers, and operating systems, ensuring seamless functionality and consistent user experience across various environments.

NFR3_CS_TC001: Verify Cross-Browser Compatibility

Steps:

- Access the application using different web browsers (e.g., Chrome, Firefox, Safari).

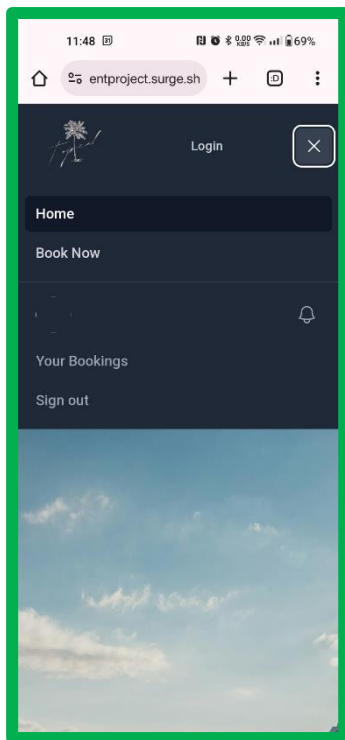


Expected Result: Application should render consistently and function correctly across all supported web browsers.

NFR3_CS_TC002: Verify Mobile Compatibility

Steps:

- Access the application using mobile devices with different screen sizes and resolutions.



Expected Result: Application should be responsive and display optimally on various mobile devices without layout issues or functionality problems.

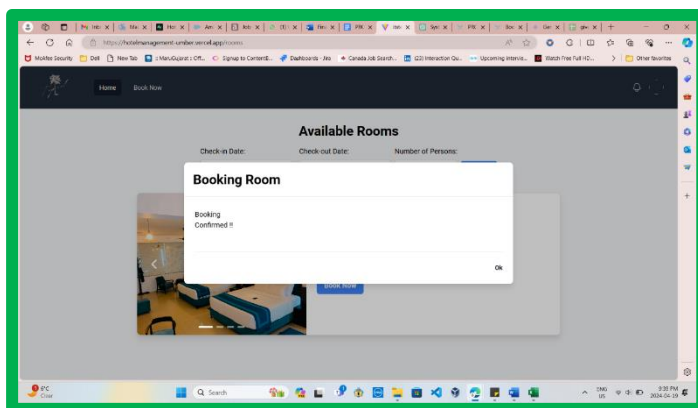
2.1.8 Functional System Testing:

Verify the functional requirements of the Hotel Booking Management System, ensuring that each feature and functionality operates according to specifications and meets the intended objectives of the system.

FR5_FS_TC001: Verify Room Booking Functionality

Steps:

- Access the room booking section.
- Select desired room type and dates.
- Confirm the booking.

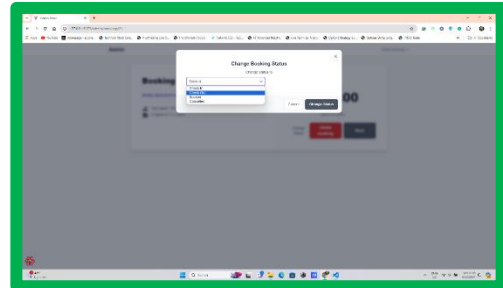
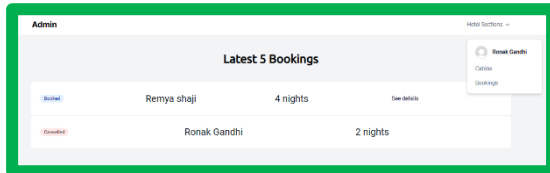


Expected Result: Booking should be successfully processed, and the user should receive a confirmation email.

FR9_FS_TC002: Verify Functionality to Modify Booking Details by admin

Steps:

- Access the booking management section.
- Modify booking details (e.g., dates, room type).



Expected Result: Changes should be reflected accurately in the booking details.

System Testing Test Cases Results

Test Case	Total Test Cases	Passes Test Cases	Failed Test Cases	Comments
Manual Test Cases	4	4	0	All Manual Test Cases are Passed
Database Test Cases	2	2	0	All Database Test Cases are Passed
Security Test Cases	2	2	0	All Security Test Cases are Passed
Performance Test Cases	2	2	0	All Performance Test Cases are Passed
Functional Test Cases	2	2	0	All Functional Test Cases are Passed
Compatibility Test Cases	2	2	0	All Compatibility Test Cases are Passed
Usability Test Cases	2	2	0	All Usability Test Cases are Passed

Unit Test

Purpose:

The primary objective of unit testing in a hotel booking system is to make certain that each system component works properly in solitude. Unit tests emphasize on the application's smallest components, which include functions or approaches, to ensure that they provide the desired outcomes when given different inputs.

Unit Test Plan

Scope:

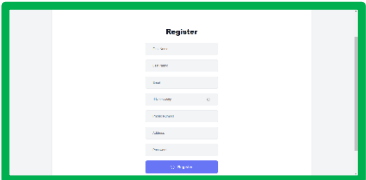
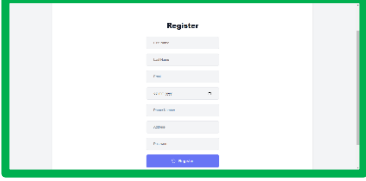
The scope of testing includes functionalities accessible to both administrators and users on the hotel booking system. This entails testing the booking procedure, such as reservation creation, modification, and cancellation; checking room availability; processing and verifying payments; managing customer accounts, which comprises the registration and authentication; and any business rules, like as limitations on reservations times or room categories. Furthermore, tests should validate the storage and management of data, including guest information and reservation history.

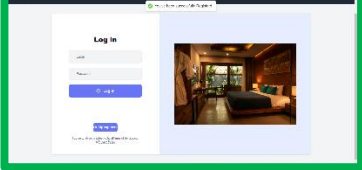
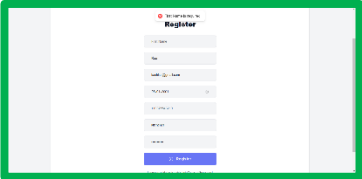
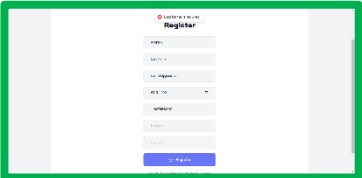
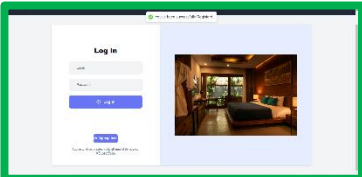
Out of Scope:



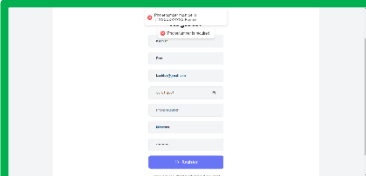

Unit testing for housekeeping management, feedback manager, billing and notification were not performed in this testing phase.


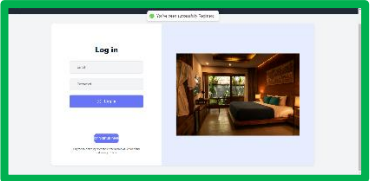
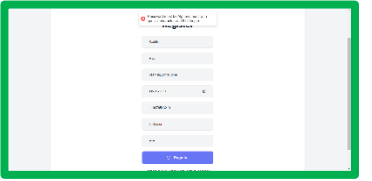
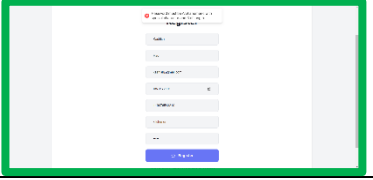
Unit Test Cases

Test Cases for Registration Module : FR1

Test Case ID	Test Case Description	Test Steps	Expected Result	Pass/Fail
FR1_UT_TC001	The registration form loads with success.	1. Go to the registration page's URL.	The registration form is shown.	Pass 
FR1_UT_TC002	The form includes all necessary fields.	1. Get the registration form open.	First name, Last name, Email, Date of Birth, Phone number, Address and Password fields are present on the form.	Pass 
FR1_UT_TC003	A valid customer registration.	1. Go to the registration page.	After a customer successfully	Pass





		<ol style="list-style-type: none"> Provide an accurate first name, last name, email id, date of birth, phone number and password fields. Hit the register button. 	registers, a confirmation message ought to appear.	
FR1_UT_TC004	First name field is empty.	<ol style="list-style-type: none"> Access the registration page. Leave the first name field unfilled. Enter accurate data in the remaining fields. Click the register button. 	There should be an error message stating that the first name field cannot remain empty.	<p>Pass</p> 
FR1_UT_TC005	Last name field is empty.	<ol style="list-style-type: none"> Access the registration page. Leave the last name field unfilled. Enter accurate data in the remaining fields. Click on the register button. 	There should be an error message indicating that the last name field cannot remain empty.	<p>Pass</p> 
FR1_UT_TC006	Validation of email field.	<ol style="list-style-type: none"> Provide a working email address. 	The email address has been verified.	<p>Pass</p> 
		<ol style="list-style-type: none"> Enter an incorrect format for an email address. 	Error message appears, email address is declined.	<p>Pass</p>

				
FR1_UT_TC007	Date of Birth field is empty.	<ol style="list-style-type: none"> 1. Access the registration page. 2. Don't fill in the date of birth section. 3. Provide precise information in the other fields. 4. Hit the register button. 	An error notification should be shown, noting that the date of birth section cannot be blank.	Pass 
FR1_UT_TC008	Phone number field is empty.	<ol style="list-style-type: none"> 1. Go to the registration page. 2. Keep the phone number field blank. 3. Type correct information into all remaining fields. 4. Tap the register button. 	An error message should be shown informing users that the phone number field cannot be left empty.	Pass 
FR1_UT_TC009	Invalid format of phone number.	<ol style="list-style-type: none"> 1. Navigate to the registration page. 2. Type an incorrect phone number (such as one that is too short or contains more than one letter) in the field of phone number. 	The incorrect phone number format should be indicated by an error message that appears.	Pass 

		<ol style="list-style-type: none"> 3. Add correct information in the remaining fields. 4. Click the register button. 		
FR1_UT_TC0010	Address field is empty.	<ol style="list-style-type: none"> 1. Visit the registration page. 2. Enter nothing in the address field. 3. Insert reliable information in the remaining fields. 4. Click on the register button. 	The address box cannot be empty, hence an error notice shown be displayed.	Pass 
FR1_UT_TC0011	Validation of password field.	<ol style="list-style-type: none"> 1. Type in a password that satisfies the requirements. 	The password has been recognized.	Pass 
		<ol style="list-style-type: none"> 2. Enter a password that is less than the required length. 	The password has been refused, and an error message is displayed.	Pass 
		<ol style="list-style-type: none"> 3. Kindly enter a password devoid of special characters. 	The password is rejected, and an error message is displayed.	Pass 

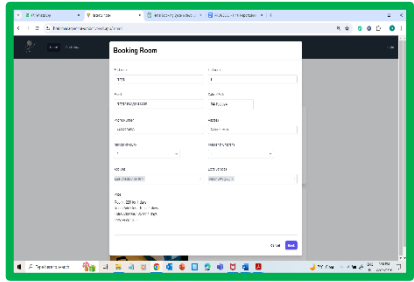
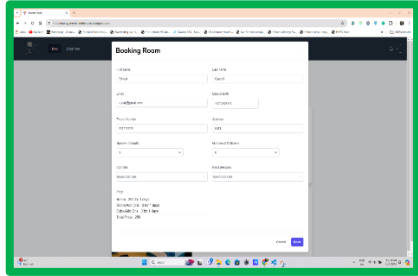
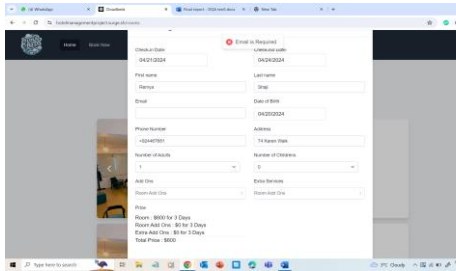
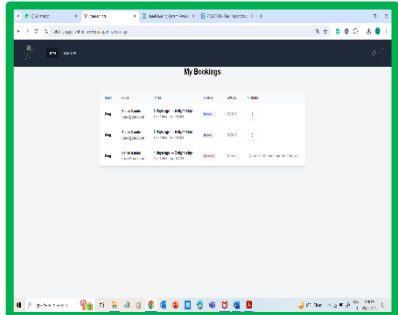
Test Cases for Login Module: FR2

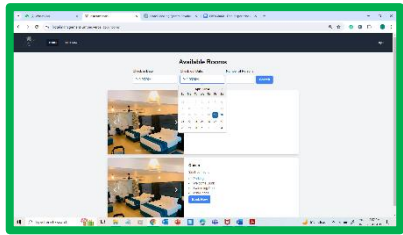
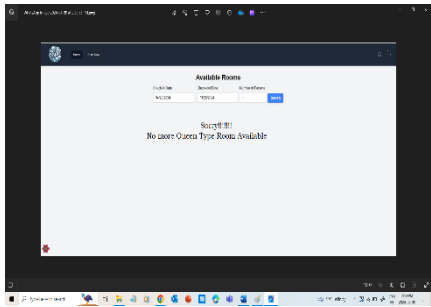
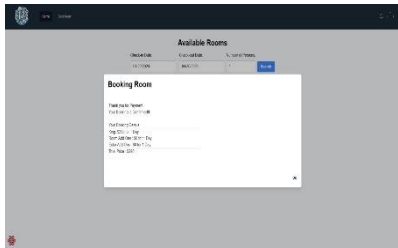
Test Case ID	Test Case Description	Test Steps	Expected Result	Pass/Fail
FR2_UT_TC001	Valid credentials for customer or administrator.	Proceed to the login page.	After successfully logging in, the user or	Pass

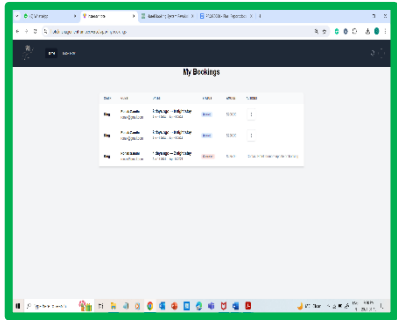
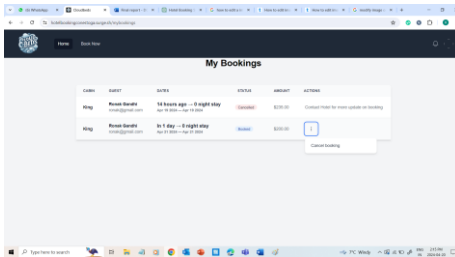
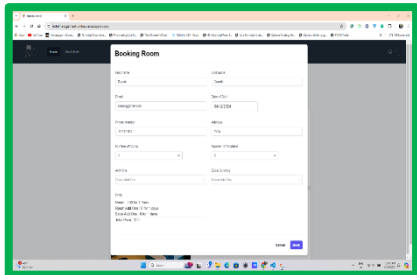
		Fill in an active email address. Fill in an authorized password. Hit the login button.	administrator should be taken to their dashboard or home page.	
FR2_UT_TC002	Invalid email id.	Go to the login page. Enter an incorrect email address. Enter in the correct password. Click on the login button.	If the email address or password entered is invalid, a warning message should appear.	Pass 
FR2_UT_TC003	Invalid password.	Navigate to the login page. Write in an active email address. Write in an incorrect password. Press the login button.	An error notification stating that the password or email address provided is incorrect ought to show up.	Pass 
FR2_UT_TC004	Email address and password fields are empty.	Head to the login page. Enter nothing in the spaces for the password and email address. Select the login button.	A warning message is required to be displayed advising that the email address and password fields cannot be null.	Pass 

Test Cases for Booking Module: FR5

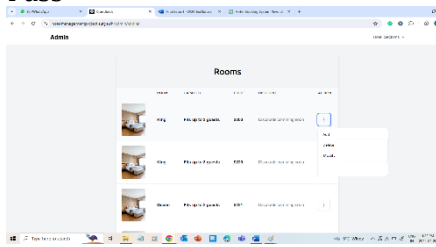
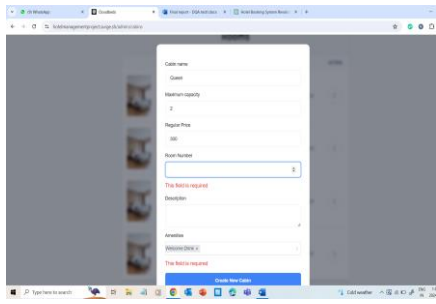
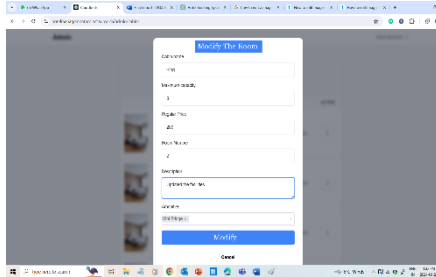
Test Case ID	Test Case Description	Test Steps	Expected Result	Pass/Fail
FR5_BM_TC001	Verify room booking for guest users	1. Navigate to booking page 2. Select desired room	Room booking is successful for guest users	Pass

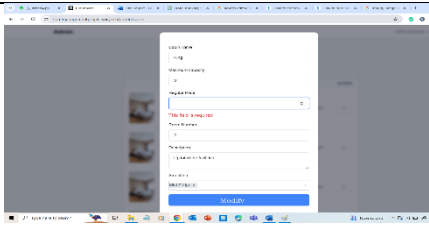
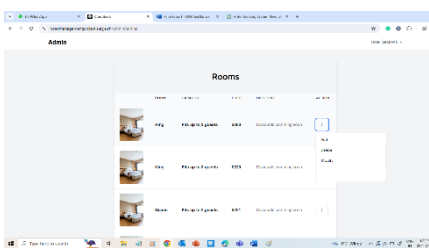
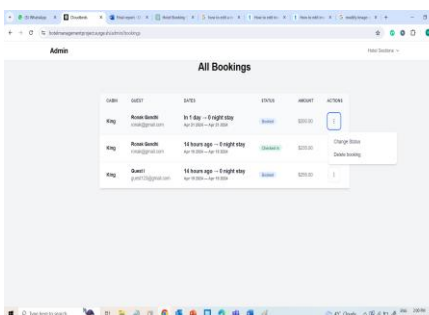
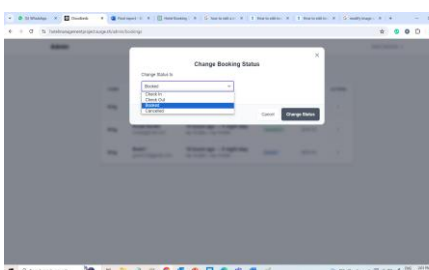
		<ol style="list-style-type: none"> 3. Enter booking details 4. Click on "Book Now" button 		
FR5_BM_TC002	Verify room booking for registered users	<ol style="list-style-type: none"> 1. Log in as registered user 2. Navigate to booking page 3. Select desired room 4. Enter booking dates user details are auto updated 5. Click on "Book Now" button 	Room booking is successful for registered users	<p>Pass</p> 
FR5_BM_TC003	Verify room booking validation for empty fields for guest users	<ol style="list-style-type: none"> 1. Navigate to booking page 2. Leave one or more required fields empty 3. Click on "Book Now" button 	Error message displayed for empty fields for guest users	<p>Pass</p> 
FR5_BM_TC004	Verify room booking validation for empty fields for registered users	<ol style="list-style-type: none"> 1. Log in as registered user 2. Navigate to booking page 3. Leave one or more required 	Error message displayed for empty fields for registered users	<p>Pass</p> 

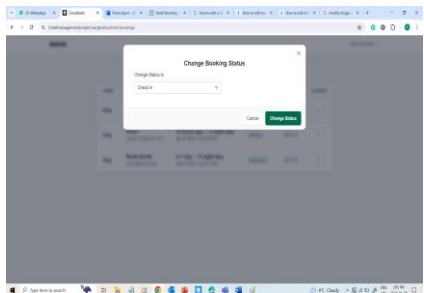
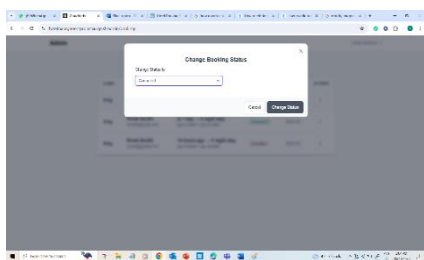
		fields empty 4. Click on "Book Now" button		
FR5_BM_TC005	Verify room booking validation for invalid date for guest users/register ed users	1. Navigate to booking page 2. Enter past date or non-date value for booking date 3. Click on "Book Now" button	Only valid dates will be given to the users to choose.	Pass 
FR5_BM_TC006	Verify room booking validation for unavailable room for guest users / registered users	1. Log in as registered user 2. Navigate to booking page 3. Select a room that is already booked 4. Enter booking details 5. Click on "Book Now" button	Error message displayed for unavailable room for registered users	Pass 
FR5_BM_TC007	Verify room booking confirmation message for guest users / registered users	1. Navigate to booking page 2. Select a room 3. Enter booking details 4. Click on "Book Now" button	Confirmation message displayed for successful room booking for guest users	Pass 

FR5_BM_TC008	Booking management functionality	<ol style="list-style-type: none"> 1. Log in as admin 2. Navigate to booking management page 3. Update booking status 4. Verify changes reflected 	Booking status is updated successfully	<p>Pass</p> 
FR6_BM_TC009	User cancellation functionality	<ol style="list-style-type: none"> 1. Log in as user 2. Navigate to booked rooms 3. Select booking to cancel 4. Confirm cancellation 	Booking is successfully canceled	<p>Pass</p> 
FR5_BM_TC010	Booking a room with add-on facilities	<ol style="list-style-type: none"> 1. Navigate to booking page 2. Select a room. 3. Choose add-on facilities 4. Enter booking details 5. Click on "Book Now" button 	Room is successfully booked with chosen add-ons	<p>Pass</p> 

Test Cases for Admin Module: FR3

Test Case ID	Test Case Description	Test Steps	Expected Result	Pass/Fail
FR3_AM_TC001	Add new room	<ol style="list-style-type: none"> 1. Log in as admin 2. Navigate to "Room Management" section 3. Click on "Add Room" button 4. Enter room details 	New room is successfully added to the system	<p>Pass</p> 
FR3_AM_TC002	Add new room with invalid data	<ol style="list-style-type: none"> 1. Log in as admin 2. Navigate to "Room Management" section 3. Click on "Add Room" button 4. Enter invalid room details 	Room addition fails with appropriate error message	<p>Pass</p> 
FR3_AM_TC003	Update room details	<ol style="list-style-type: none"> 1. Log in as admin 2. Navigate to "Room Management" section 3. Select a room to update 4. Modify room details 	Room details are successfully updated	<p>Pass</p> 
FR3_AM_TC004	Update room details with invalid data	<ol style="list-style-type: none"> 1. Log in as admin 2. Navigate to "Room Management" section 	Room update fails with appropriate error message	<p>Pass</p>

		3. Select a room to update 4. Enter invalid room details		
FR3_AM_TC005	Delete room	1. Log in as admin 2. Navigate to "Room Management" section 3. Select a room to delete 4. Confirm deletion	Selected room is successfully deleted from the system	<p>Pass</p> 
FR3_AM_TC006	Cancel booking	1. Log in as admin 2. Navigate to "Booking Management" section 3. Select a booking to cancel 4. Confirm cancellation	Selected booking is successfully canceled	<p>Pass</p> 
FR3_AM_TC007	Modify booking details	1. Log in as admin 2. Navigate to "Booking Management" section 3. Select a booking to modify 4. Update booking details	Booking details are successfully updated	<p>Pass</p> 
FR3_AM_TC008	Approve booking	1. Log in as admin 2. Navigate to "Booking Management" section	Selected booking is successfully approved	<p>Pass</p>

		3. Select a pending booking 4. Approve the booking		
FR3_AM_TC009	Reject booking	1. Log in as admin 2. Navigate to "Booking Management" section 3. Select a pending booking 4. Reject the booking	Selected booking is successfully rejected	Pass 

Conclusion

The Hotel Booking Management System project has successfully transformed reservation processes in the hospitality industry, offering a comprehensive platform that addresses critical pain points and enhances the overall booking experience. Through meticulous planning, development, and extensive testing, the system now stands as a robust and user-centric solution, streamlining operations, optimizing resource allocation, and fostering customer satisfaction. By prioritizing intuitive user interfaces, robust booking management functionalities, and stringent security measures, the system ensures a seamless and secure experience for both customers and administrators alike, affirming its reliability and effectiveness. Throughout the development lifecycle, various testing phases were conducted to validate the system's functionality, performance, and security, ensuring compliance with all specified design and functional requirements. These rigorous testing procedures have instilled confidence in the system's capabilities, preparing it for deployment in real-world scenarios. Moving forward, the project remains committed to continuous improvement, driven by user feedback and industry trends. Embracing an iterative approach to development, the Hotel Booking Management System will continue to evolve, adapt, and innovate, driving operational efficiency and excellence in reservation management for years to come.

Recommendation

Based on the project outcomes, the following recommendations are made for future development:

Performance Optimization: Continue to monitor and optimize system performance, particularly focusing on handling high traffic volumes to maintain a smooth user experience.

Security Enhancements: Implement additional security measures to address vulnerabilities and enhance data protection, based on the findings from security testing.

Feature Expansion: Consider the development of B2C functionalities and advanced analytics features, as these areas were identified as out of scope in the initial phase but could provide significant value in future updates.

Continuous Testing: Maintain a rigorous testing regime, incorporating automated and manual testing strategies to ensure ongoing reliability and to address any new issues promptly.

References

1. Fowler, M. (2004). Mocks Aren't Stubs. Retrieved from:
<https://martinfowler.com/articles/mocksArentStubs.html>
2. Meszaros, G. (2007). Unit Test Patterns: Refactoring Test Code. Addison-Wesley Professional.
3. Freeman, S., & Pryce, N. (2009). Growing Object-Oriented Software, Guided by Tests. Addison-Wesley Professional.