Haramaya University

Building the Basis for Development

College of Computing and Informatics

Department of Software Engineering

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Document for

FOR Feven HOTEL/RESTAURANT MANAGEMENT SYSTEM

Group members	ID
1. ABENEZER YOHANNES	861/13
2. EDEN BIRHANU	882/13
3. GETHAUN TAMIRAT	421/12
4. HEAVEN ENDRIS	889/13
5. NIYA MURAD	901/13

Advisor name

Mr. Dita A.

Date:

Signature

Submission date: MAY 19 , 2025

Submitted to:DEPARTMENT OF SOFTWARE ENGINEERING

Table of content

Table 1 : Table of content

Lists of Tables pages

Lists of Figures

Definition, Abbreviation and Acronyms

1. Software Requirements Specification (SRS)
2. Hotel Management System (HMS)
3. functional requirements (REQ)
4. Non-functional requirements (NFR)
5. Quality Assurance (QA)
6. BUISSNES rule (BR)
7. Requirements (REQ)
8. Use Case (UC)
9. Alternative Flows (AF)
10. Data Validation Rules (DR)
11. Mean Time Between Failures (MTBF)
12. cross-site scripting (XSS)
13. cross-site request forgery (CSRF)

CHAPTER ONE: INTRODUCTION

1.1 Document Scope

This Software Requirements Specification (SRS) document describes the detailed requirements for the design and implementation of a Web-Based Hotel Management System (HMS). The document outlines the system's functionalities, constraints, interfaces, and performance requirements. The system aims to streamline and automate various hotel operations including room reservations, guest registration, check-in/check-out processes, billing and payments, staff management, inventory control, and reporting.

This SRS is structured to provide a logical flow of information starting from general descriptions to detailed system requirements. The sequence of sections allows each type of stakeholder to focus on the components relevant to their role. Developers should start from Section 2 and Section 3 for implementation details, while project evaluators and supervisors may focus on the introduction and overall system architecture for a broader understanding of the project objectives.

1.2 Document Purpose

This Hotel Management System Software Requirement Specification (SRS) main objective is to provide a base for the foundation of the project. It gives a comprehensive view of how the system is supposed to work and what is to be expected by the end users. Client's expectation and requirements are analyzed to produce specific unambiguous functional and non-functional requirements, so they can be used by development team with clear understanding to build a system as per end user needs.
This SRS for HMS can also be used for future as basis for detailed understanding on how project was started. It provides a blueprint to upcoming new developers and maintenance teams to assist in maintaining and modifying this project as per required changeability.

Specifically, the document:

- Defines the core functionalities the system must support, such as room booking, customer and staff management, and report generation.
- Serves as a reference for future system maintenance, updates, and scalability enhancements.
- Establishes a foundation for verification and validation through testing processes.
- Ensures traceability of requirements throughout the software development lifecycle.

1.3 Document Conventions

This document adheres to the Software Engineering Project standards provided by the College of Computing and Informatics, Haramaya University. The following conventions are used throughout the document to maintain consistency and clarity:

Formatting Conventions:

- **Bold text** is used for headings and section titles.
- *Italic text* emphasizes key terms or important notes.

Naming Conventions:

- Functional requirements are expressed using the format: *"The system shall [do something]"*.
- Actor names (e.g., *Receptionist*, *Administrator*, *Customer*) are capitalized.
- All diagrams are labeled with descriptive titles and figure numbers.

Requirement Identification:

- Each requirement is assigned a unique identifier (e.g., REQ-001 for functional requirements, NFR-001 for non-functional requirements) for traceability.

1.4 Intended Audience

The intended audience of this document includes, but is not limited to:

- **Software Developers**: To understand the complete functionality and technical specifications of the system for accurate development.

- **Project Advisor**: To monitor the project's progress and ensure adherence to academic and technical standards.**Examiners and Reviewers**: To assess the project for technical completeness, correctness, and relevance to real-world needs.

- **Documentation Writers**: To prepare user manuals and help documentation based on the requirements.

- **End Users** (e.g., hotel managers, receptionists): To understand the scope and capabilities of the system, though they are more likely to read user documentation derived from this SRS.

1.5 Team Organization

The Web-Based Hotel Management System is developed by a dedicated team of final-year Software Engineering students. The team is organized as follows:

Member Name	Role	Responsibilities
NIYA & ABENEZER	Project Lead & Full-stack Developer	Lead the project, system architecture, database design, and full-stack development
EDEN BIRHANU	Backend Developer	Design and implement the server-side logic, REST APIs, and database integration
GETHAUN TAMIRAT	Frontend Developer	Develop the user interface using modern web technologies and ensure responsiveness
HEAVEN ENDRIS	Project management	Write and execute test plans, perform functional and non-functional testing
Mr. Dita A.	Project Advisor	Provide academic supervision, feedback, and ensure the project meets curriculum standards

CHAPTER TWO: SYSTEM DESCRIPTION

2.1 SYSTEM OVERVIEW

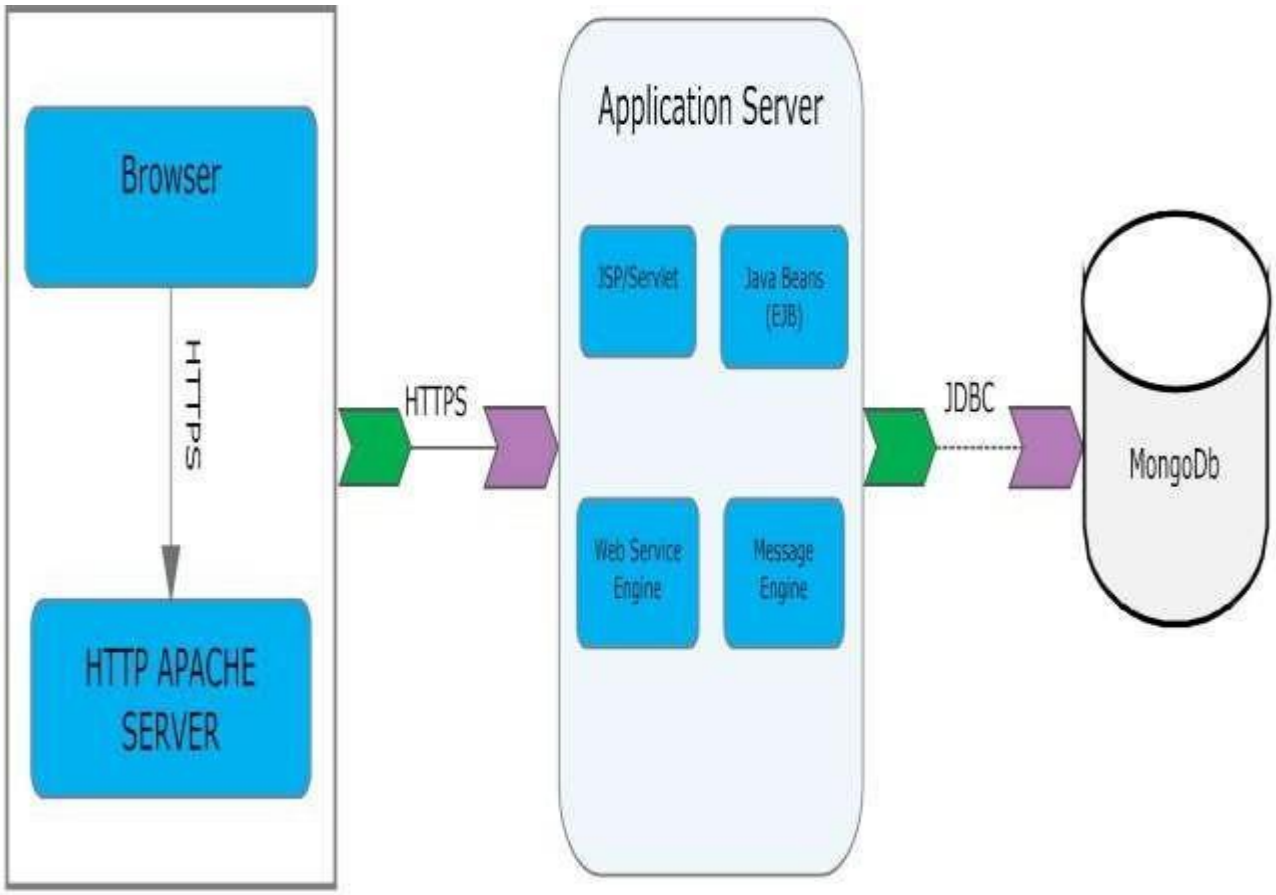
This document details the software requirements for a web-based management system specifically designed for Feven Hotel/Restaurant. The system's development is motivated by the operational challenges currently faced by Feven Hotel/Restaurant, which include inefficient manual processes for managing reservations, orders, inventory, and customer information. The general objective is to develop and implement a system that streamlines operations, improves efficiency, and enhances customer satisfaction. This project is significant because it aims to provide Feven Hotel/Restaurant with a modern and efficient tool, reduce errors, save time, improve staff productivity, and offer greater convenience for customers through online services. Furthermore, it serves as a practical example of a web-based management system suitable for small businesses.

The system will provide a centralized, user-friendly platform accessible via the internet. It is a new, self-contained software product intended to replace or significantly improve upon the current manual system, which relies on phone calls and spreadsheets. The proposed system seeks to overcome problems associated with handling large physical files, calculation errors, and other tasks currently performed manually.

At a high level, the system's workflow involves customers interacting with the online interface for reservations and ordering, while staff and administrators utilize the system for managing these activities, updating system data, and generating reports. The core functionalities encompass managing online table reservations, online food/drink orders (for both dine-in and takeout), maintaining menu details, tracking inventory levels, managing customer information, and generating essential sales reports. A high-level context diagram, as suggested by the SRS outline, would visually represent the system boundary and its major interactions with external entities such as customers, staff, and potentially external systems (like a single payment gateway).

The system will be developed using a specific technology stack including HTML, PHP, React, Node.js, and MySQL. This combination of technologies is considered suitable for developing a scalable and reliable web application.

Figure 1: high level context diagram of the system



2.2 System Scope

The scope of the Feven Hotel/Restaurant Management System is defined by the functionalities and boundaries agreed upon for this project. The system is designed to automate and streamline major operations for Feven Hotel/Restaurant. It will be a web-based system, meaning it will be accessible via the internet through standard web browsers. The purpose is to simplify daily processes, handle services quickly to enhance customer service, and improve efficiency by reducing the reliance on manual file handling, increasing ease of use, safety, and information retrieval compared to the existing manual system.

The system will explicitly include the following key functionalities:

* Online Table Reservations: Allowing customers to book tables online.
* Online Ordering: Enabling customers to place orders for both dine-in and takeout.
* Menu Management: Providing administrators/staff with tools to manage the food and drink menu.
* Order Management: Supporting staff in processing, tracking, and managing customer orders.

* Inventory Management: Allowing tracking and management of food and beverage inventory.
* Sales Reporting: Generating reports based on sales data.
* Customer Management: Storing and managing customer information and order history.

The system will not include certain functionalities within the scope of this project:

* Integration with third-party accounting software.
* Comprehensive payment gateway integration; it will be limited to a single, specified payment gateway.
* A dedicated native mobile application (access will be via web browsers, potentially including mobile-friendly web design).

This defined scope provides the boundaries of the project, clarifying what will be delivered within the allocated time and resources. Requirements that might be delayed until future versions of the system are not included in this scope definition.

2.3 User Characteristics

There are 3 user Levels in our Hotel Management System:
A. Hotel Manager
B. Receptionist
C. Customers

Hotel Manager
The Manager at Feven Hotel/Restaurant will have every access to the system. They are solely responsible for managing hotel resources and staff. Given that Feven currently relies on manual processes leading to inefficiencies and difficulties in tracking performance, the Manager will significantly benefit from the system's ability to view any report, such as financial reports, customer information, booking information, and room information. This access allows them to analyze data and take decisions accordingly to address problems like potential revenue loss and reduced staff productivity. The role requires experience in managing a hotel and a base knowledge of database and application server, enabling them to leverage the system to streamline operations for both the hotel and restaurant aspects of the business.

Receptionist
The Hotel Receptionist's primary purpose is to provide quality customer service. Their access level is less than the Manager's. For Feven, which currently takes reservations primarily via phone, leading to potential errors and double bookings, the Receptionist will use the system to manage booking details. This includes the ability to search for availability of rooms and likely handle table reservations as the system covers the restaurant as well. They can add the customer, confirm the booking, and update the booking details. The Manager would prefer a Receptionist with good communication skills, command over English, and basic IT knowledge to effectively use the new system for managing guests and

reservations. The system aims to improve efficiency and ease of use compared to the previous manual system.

Customer
Customers are a vital part of the system, especially as Feven is implementing a web-based platform. Customers will have access to view vacant room information and price range. They should be able to confirm the booking and cancel it if necessary, directly addressing the manual reservation drawbacks. The proposed system for Feven includes features like online ordering and table reservations, indicating customers will also interact with the system for these restaurant services. Customers should at least be capable of using the web UI interface. They also have access to a customer service desk portal to forward their inquiry.

The system is designed with different authorization levels to ensure that different user types have appropriate access to system functionalities, protecting data and preventing unauthorized operations

2.4 BUISSNES RULE

This section lists the key business rules that govern the operations of Feven Hotel/Restaurant and must be enforced by the system. These rules represent policies, guidelines, or constraints that emerge from the business processes. Adherence to these rules is essential for the system to meet the objectives of streamlining operations and improving efficiency.

Rule ID	Description	Impacted Area/Requirements
BR-001	Table Availability: Table reservations must be based on the available seating capacity and pre-defined time slots. The system must prevent overbooking a table at any given time.	Online Table Reservations, Availability Search, Booking Confirmation
BR-002	Room Availability: Room bookings must be based on the current room status (Vacant, Occupied, Maintenance) and available room types. The system must	Room Booking (if applicable based on full HMS scope), Availability Search, Booking Confirmation

	prevent double booking rooms.	
BR-003	Reservation Lead Time: Online table reservations must be made at least 2 hours in advance. Future bookings are limited to 30 days	Online Table Reservations
BR-004	Order Association: Dine-in orders placed through the system must be associated with a specific table number. Takeout orders must be associated with a customer profile.	Online Ordering (Dine-in), Online Ordering (Takeout), Order Management, Customer Management
BR-005	Menu Pricing: All menu items must have a defined price. Prices displayed to customers must include any applicable taxes or service charges as per restaurant policy.	Menu Management, Online Ordering, Sales Reporting
BR-006	Order Status Workflow: Orders must progress through a defined set of statuses (e.g., Received, Preparing, Ready for Pickup/Serve, Completed, Cancelled). Only staff roles with appropriate privileges can change the status	Order Management, Staff Interface
BR-007	Inventory Deduction: Upon completion of an order (or relevant status transition), the system must deduct the corresponding	Order Management, Inventory Management

	ingredients/items from the inventory based on pre-defined recipes or item quantities.	
BR-008	Low Stock Alert: When the quantity of an inventory item drops below a pre-defined minimum threshold, the system must generate an alert or notification	Inventory Management, Reporting
BR-009	Menu Updates: Only users with the "Hotel Manager" role shall have the ability to add, edit, or remove menu items, including updating prices and availability status.	Menu Management, User Classes and Characteristics (Access Control)
BR-010	Customer Data Access: Customer information, including contact details and order history, is confidential and accessible only to authorized staff roles (Manager, Receptionist) for legitimate business purposes (e.g., managing bookings, fulfilling orders). Customers can access their own profile and history.	Customer Management, User Classes and Characteristics (Access Control), Security Requirements
BR-011	Reporting Access: Financial and sales reports are confidential and restricted to the "Hotel Manager" role for business analysis	Sales Reporting, User Classes and Characteristics (Access Control), Security Requirements

	and decision-making	

Table 2: lists of Business rules of systems

2.5 Operating Environment

The operating environment defines the technical infrastructure on which the Hotel Management System will run after it is completed. This system is specified as a web-based application, which determines the nature of its operating environment.

* **Server Environment:** The backend components of the system will reside on a server. This server environment is capable of hosting applications built with Node.js and PHP. It will also host the MySQL database used for storing all system data. The server requires a suitable Operating System (e.g., Linux, Windows Server) that supports the chosen technologies. Necessary web server software is also required to handle incoming web requests. The server's hardware (CPU, RAM, Storage) must be sufficient to handle the expected load, although specific performance requirements may be detailed in Chapter 3.

* **Client Environment:** End users (Customers, Staff, Administrators/Managers) will access the system through standard web browsers. The frontend, built with React and HTML, will run within these browsers. The client environment requires a device capable of running a compatible web browser (e.g., desktop computer, laptop, tablet, smartphone). Minimum client hardware requirements are generally low, primarily needing a stable internet connection and a functional display. Users should use relatively modern browsers to ensure compatibility with modern web technologies.

* **Network Environment:** Accessing the web-based system necessitates a broadband internet connection for both the server (for availability) and client devices (for access). A local network within the hotel/restaurant may be used for staff/administrator access to local resources or the server if hosted on-premises, but internet access is required for external customer access and potentially for features like email notifications.

* **Additional Hardware/Software:** The system may require a printer for generating physical outputs such as customer bills, order tickets for the kitchen, or sales reports. While not part of the core system, the operating environment includes these peripheral dependencies. The client (Feven Hotel/Restaurant) is assumed to afford necessary software and hardware costs associated with the operating environment.

The performance of the system will depend significantly on the performance of the underlying hardware and software components in this operating environment

2.6 DESIGN AND IMPLEMENTATION CONSTRAINTS

The system is constrained by the internet access. Since the system fetches from the database via internet access is the main constraint for the system.
The system uses centralized database the other constraint might be the capacity of the database it may force to queue incoming requests therefore increase the time it takes to fetch data.
The government trust on ICT is also a constraint that affects the system. Since use of ICT applications in Ethiopia is a new emerging technology the trust of the government may be in question.
The capability of individuals to use the system is a constraint that affects the system hence individuals should able to know the basic of computer and internet.

Other important constraint lists as follow :

I. **Memory:** System will have only limited space of data server.
II. **Language Requirement:** Software must be only in English.
III. **Budget Constraint:** Due to limited budget, HMS is intended to very simple and, just for basic functionalities. UI is going to be very simple.
IV. **Implementation Constraint:** the platfrom is limited to Web based only.
V. **Reliability Requirements:** System should sync frequently to backup server in order to avoid the data loss during failure, so it can be recovered

2.7 Assumptions and Dependencies

The following assumptions and dependencies are relevant to the Feven Hotel/Restaurant Management System project:
* **Reliable Internet Access**: It is assumed that all primary users (Customers, Staff, Administrators/Managers) will have reliable access to the internet to utilize the web-based system. The system's core functionality depends entirely on network connectivity.

* **Server Environment Availability and Stability:** It is assumed that a suitable server environment capable of hosting the /PHP application and MySQL database will be available, properly configured, and reliably maintained. The system depends on the server's uptime and performance.

* **Availability of Necessary Hardware**: It is assumed that the necessary hardware, including computers for staff and administrators, and a printer for printing outputs like bills or reports, will be available at the Feven Hotel/Restaurant premises.

* **Stability and Functionality of Chosen Technologies:** The project depends on the stability, reliability, and continued support for the chosen open-source technologies: React,

Node.js, PHP, and MySQL. Issues or vulnerabilities in these components could impact the project.
* **Data Integrity:** The system depends on the ability to maintain data integrity and consistency within the MySQL database. Proper database design and application logic are required to ensure this.

* **Payment Gateway Reliability:** The system depends on the reliability and functionality of the single chosen payment gateway for processing online payments.

* **Availability of Staff for Training/Feedback:** It is assumed that key staff members and administrators will be available for providing feedback during development and for training upon deployment.

These factors represent conditions that the project relies on; deviations could affect the project's progress and outcome.

CHAPTER THREE: SYSTEM REQUIREMENTS

3.1 Functional Requirements

A. User Management

This feature encompasses functionalities related to managing user accounts and access to the system, particularly for guests and potentially staff.

* **REQ-3.1.1:** The system shall allow new users (guests) to register for an account.

Details: Registration requires providing details such as full name, contact information (email address, phone number), and creating a password. The system should perform validity checks on inputs, such as email format and password complexity.

Response: Upon successful registration, the system shall create a new user account and notify the user of successful creation.

* **REQ-3.1.2:** The system shall allow registered users (guests and staff) to log in using their credentials.

Details: Users provide a username (e.g., email address) and password. The system shall authenticate the credentials against stored user data.

Response: Upon successful authentication, the system shall grant access appropriate to the user's role. Upon failed authentication (e.g., incorrect credentials), the system shall display an error message.

.

* **REQ-3.1.3:** The system shall allow administrators (or designated staff) to manage user accounts, including creating, viewing, updating, and deactivating staff accounts.

Details: This includes assigning roles (e.g., Receptionist, Housekeeping, Manager) with specific permissions.

Response: The system shall reflect the changes in the user database and enforce the assigned roles and permissions.

B. Reservation Management

This feature covers the core functionality of booking, viewing, modifying, and canceling room reservations.

* **REQ-3.1.4:** The system shall allow users to search for room availability based on check-in date, check-out date, room type, and number of guests.

Details: The system shall query the room and reservation data to identify available rooms that match the criteria for the specified date range.

Response: The system shall display a list of available rooms, including room type, capacity, price, and amenities. If no rooms are available, the system shall inform the user.

* **REQ-3.1.5:** The system shall allow authenticated users (guests) to create a new reservation.

Details: The user selects a room from the search results (REQ-3.1.5) and provides reservation details, potentially including guest information (if different from the booking user) and payment information. The system shall validate input data and check room availability again before confirming the booking.

Response: Upon successful creation and potential payment processing (see REQ-3.1.12), the system shall generate a reservation confirmation and store the reservation details.

* **REQ-3.1.6:** The system shall allow guests and staff to view existing reservations.

Details: Guests can view their own reservations. Staff can view all reservations based on search criteria (e.g., guest name, date range, reservation ID).

Response: The system shall display reservation details, including guest information, booking dates, room details, status, and payment information.

* **REQ-3.1.7:** The system shall allow authorized users (staff) to modify existing reservations.

Details: Staff can change reservation dates, assigned room (within the same type, if available), number of guests, or guest information. Changes may be subject to availability checks and policy constraints (see Section 3.5 Business Rules).

Response: The system shall update the reservation record and confirm the changes.

* **REQ-3.1.8:** The system shall allow authorized users (staff) to cancel existing reservations.

Details: Staff can cancel a reservation. Cancellation may be subject to policy constraints (see Section 3.5 Business Rules).

Response: The system shall update the reservation status to 'Canceled' and handle associated financial transactions (e.g., refunds), potentially triggering functional requirements related to billing.

C. Room Management

This feature focuses on functionalities related to the physical rooms and their status within the hotel.

* **REQ-3.1.9:** The system shall allow staff to manage room details.

Details: This includes adding new rooms, updating existing room information (room number, type, capacity, price, status), and removing rooms from the system inventory. The system shall validate input data for accuracy and completeness.

Response: The system shall update the room inventory and make the information available for room search and assignment processes.

* **REQ-3.1.10:** The system shall allow staff to assign a room to a guest during check-in.

Details: Given a reservation, the system shall present available rooms of the reserved type for assignment. Staff select a specific room.

Response: The system shall link the selected room to the reservation record and update the reservation status to 'Checked-in'.

* **REQ-3.1.11:** The system shall allow staff to manage room status (e.g., Available, Occupied, Under Maintenance, Needs Cleaning, Clean).

Details: Staff can manually update the status of a room. The system should automatically update the status based on check-in/check-out and housekeeping actions.

Response: The system shall update the room status record, affecting availability search results and housekeeping assignments.

D. Billing and Payment

This feature handles generating invoices and processing payments for reservations and potentially other services.

REQ-3.1.12: REQ-3.1.17: The system shall integrate with the Chapa payment gateway to securely process online payments made by guests.

Details: The system shall initiate payment sessions using Chapa's API, redirect customers to Chapa's hosted payment page, and handle callbacks or webhooks to update payment statuses in real-time.

Response: Upon successful payment, the reservation or order status shall be updated to reflect confirmation. Failed or canceled transactions will return the user to the payment screen with an error message.

E. Point of Sale / Order Management

This feature, if included based on the project scope, handles ordering and billing for additional services like restaurant or room service.

* **REQ-3.1.13:** The system shall allow staff to manage inventory of items sold via POS (e.g., restaurant ingredients, minibar items).

Details: This includes adding, updating, and removing inventory items, tracking stock levels, and potentially setting low-stock alerts.

Response: The system shall update the inventory records and provide reports on stock levels.

F. Housekeeping Management

This feature supports managing the cleaning status of rooms.

* **REQ-3.1.14:** The system shall automatically update room status to 'Needs Cleaning' upon guest check-out.

Details: This occurs immediately after the check-out process is completed.

Response: The system shall update the room status record.

* **REQ-3.1.15:** The system shall allow housekeeping staff to view a list of rooms requiring cleaning.

Details: The list should include room number and current status. It may also show priority or time since check-out.

Response: The system shall display the list of rooms needing attention.

* **REQ-3.1.16:** The system shall allow housekeeping staff to update a room's status to 'Clean' once cleaning is completed.

* **Details**: Staff can select a room from their list and mark it as clean.

* **Response**: The system shall update the room status record, making the room available for new check-ins.

3.2 USE CASE MODEL

A) USE CASE DIAGRAM

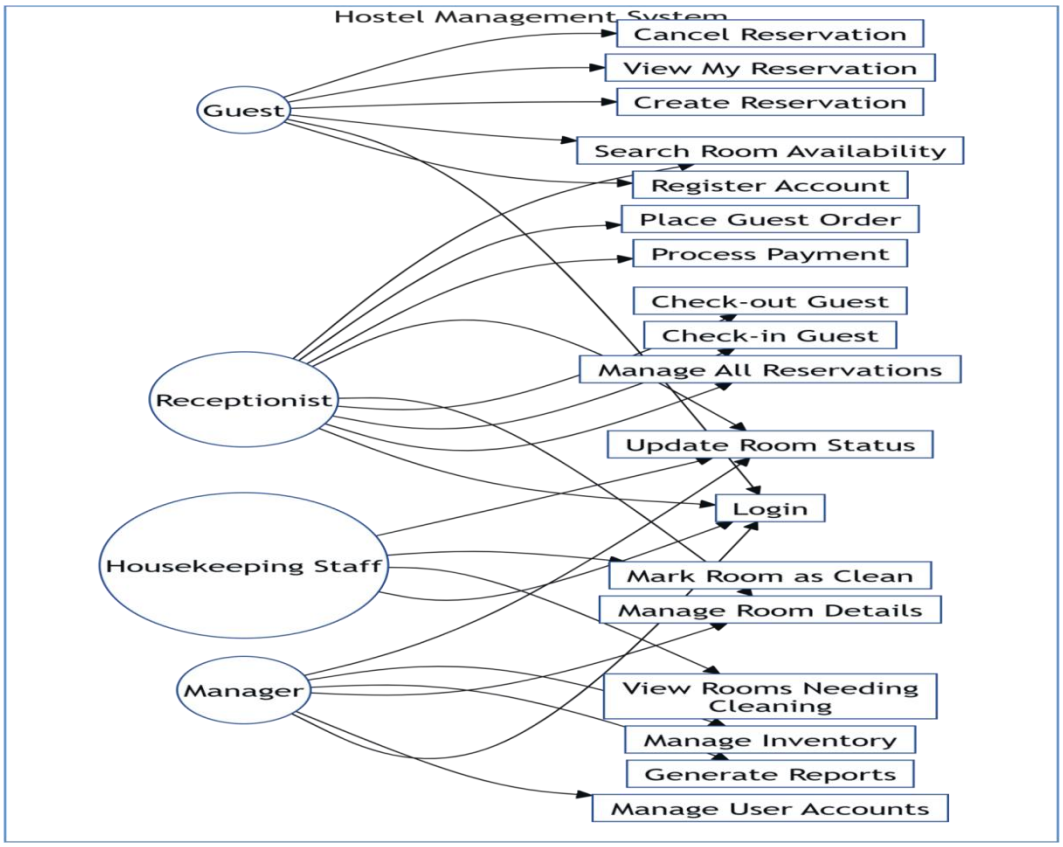


Figure 2: use case diagram

B) Use Case Description

ᴜse Case 1: Create Reservation

Use Case ID: UC-001

Use Case Name: Create Reservation

Actor(s): Guest, Receptionist

Goal in Context: The user successfully reserves a room for a specified period.

Scope: The Hotel Management System.

Preconditions:

The user is logged into the system (for Guests) or is a logged-in Receptionist.

Room types and prices are configured in the system.

The system is operational and accessible.

Postconditions:

Success: A new reservation record is created in the system, linked to the user/guest and the reserved room type. If payment was processed as part of booking, a payment record is also created. A confirmation (receipt/message) is generated and provided to the user.

Failure: No reservation record is created. The user is notified of the reason for failure.

Primary Success Scenario (Guest):

The Guest initiates the process to make a reservation, typically by navigating to the online booking interface.

The System prompts the Guest to enter search criteria (e.g., Check-in Date, Check-out Date, Room Type preference, Number of Adults, Number of Children).

The Guest enters the criteria and submits the search request.

The System searches for available rooms matching the criteria for the specified dates by querying the room and reservation data.

The System displays a list of available rooms that match the search, including room type name, capacity, price per night, amenities, and potentially images. If multiple rooms of the same type are available, it indicates the quantity.

The Guest selects a room type from the list and indicates they wish to proceed to book this type.

The System prompts the Guest to confirm reservation details (dates, room type, number of guests) and provide guest information (e.g., name, contact email, phone number), especially

if the booking user is different from the primary guest, or if the user account lacks complete details.

The Guest confirms the details and provides the required guest information. The system may validate the format of contact information.

The System prompts the Guest for payment information (e.g., credit card number, expiry date, CVV, billing address) to secure or prepay the reservation.

The Guest enters payment information and submits it.

The System validates the payment information (e.g., format, validity) and processes the payment via a secure payment gateway.

Upon successful payment processing, the System confirms successful payment and creates the reservation record in the database, associating it with the guest, room type, dates, and payment details.

The System generates and displays a reservation confirmation to the Guest, including a unique reservation ID, summary of details, and confirmation of payment. This confirmation may also be sent via email.

Alternative Flows:

AF-1.1: No Rooms Available:

Steps 4-5 (System searches and displays results): If, after searching, no rooms match the exact criteria (dates, type, capacity), the System informs the Guest that no rooms are available for those dates or criteria and may suggest alternative dates or room types if possible. (Go back to Step 2 to search again or the Guest can choose to end the Use Case).

AF-1.2: Room Becomes Unavailable During Booking:

Steps 6-11 (Guest selects room and provides info): If the selected room type becomes fully booked by another user or process between the time the Guest saw the availability list (Step 5) and when they attempt to confirm or pay (Step 11), the System informs the Guest that the room is no longer available and re-displays the currently available options or suggests alternative dates. (Go back to Step 5 or the Guest can choose to end the Use Case).

AF-1.3: Invalid Input:

Steps 3, 8, 10 (User input steps): If the Guest provides invalid input (e.g., incorrect date format, missing required field, invalid payment details format), the System displays a

specific error message indicating which input is invalid and prompts the Guest to correct it. (Return to the specific input step).

AF-1.4: Payment Failure:

Step 11 (Process Payment): If payment processing fails (e.g., insufficient funds, invalid card number, payment gateway error), the System informs the Guest of the payment failure reason and allows them to try again with different information or cancel the process. (Go back to Step 9 to re-enter payment or the Guest can choose to end the Use Case).

AF-1.5: User is not logged in (Guest):

Step 1 (Initiate reservation): If the Guest initiates the online reservation process but is not currently logged in, the System may prompt them to log in or register for an account before proceeding with entering detailed booking information. (Precede with Login Use Case (UC-00?) or Register Use Case (UC-00?)).

AF-1.6: User Skips Optional Fields:

Steps 8 (Guest information) or subsequent steps: If certain fields prompted by the system are marked as optional, the Guest may skip them. The System proceeds as long as mandatory fields are completed.

Primary Success Scenario (Receptionist):

The Receptionist initiates the process to make a reservation for a guest, usually via a dedicated staff interface.

The System prompts the Receptionist to enter search criteria (e.g., Check-in Date, Check-out Date, Room Type preference, Number of Guests).

The Receptionist enters the criteria based on the guest's request and submits the search.

The System searches for available rooms matching the criteria for the specified dates.

The System displays a list of available rooms with details and prices to the Receptionist.

The Receptionist selects a room type and proceeds, confirming the choice with the guest.

The System prompts the Receptionist to enter Guest information (Name, Contact Email, Phone, Address, etc.) and confirm reservation details. The Receptionist may also add notes or special requests provided by the guest.

The Receptionist enters the Guest and reservation details. The System may check if a guest profile already exists and prompt to link the reservation.

The System prompts for payment information based on the hotel's policy (e.g., deposit required, full prepayment, payment on arrival).
The Receptionist enters payment information if required at the time of booking, or marks the payment as deferred.
The System validates input and creates the reservation record in the database. It may process payment via a gateway if applicable (see REQ-3.1.14).
The System confirms reservation creation and displays the reservation details, including the new reservation ID, to the Receptionist.
Alternative Flows (Receptionist):
AF-1.1 (Receptionist): No Rooms Available: Similar to the Guest scenario, if no rooms match, the System informs the Receptionist. (Go back to Step 2 or end Use Case).
AF-1.2 (Receptionist): Room Becomes Unavailable During Booking: Similar to the Guest scenario, the System notifies the Receptionist if the selected room type becomes unavailable. (Go back to Step 5 or end Use Case).
AF-1.3 (Receptionist): Invalid Input: Similar to the Guest scenario, the System prompts for correction if the Receptionist enters invalid data. (Return to the input step).
AF-1.4 (Receptionist): Payment Failure: Similar to the Guest scenario, if payment fails, the System informs the Receptionist. (Go back to Step 9 or handle via AF-1.7 or end Use Case).
AF-1.6 (Receptionist): Guest Already Exists:
Step 8 (Enter Guest information): If the System recognizes the Guest details entered (e.g., by name, email, or phone number) as matching an existing profile in the system, it may prompt the Receptionist to link the new reservation to the existing guest profile instead of creating a new one. The Receptionist confirms or creates a new guest profile.
AF-1.7 (Receptionist): Payment Deferred:
Steps 9-10 (Payment): If the hotel policy or guest request allows for payment at a later time (e.g., on arrival), the Receptionist skips the payment processing step. The System creates the reservation record, and the Post condition is modified to reflect that payment is pending.
AF-1.8 (Receptionist): Apply Discount or Special Rate:
Steps 7-11 (Details/Payment): The System may allow the Receptionist to apply a discount code or select a special corporate/negotiated rate during the booking process, which affects the Estimated Total Cost . The System recalculates the cost.

Use Case 2: Check-in Guest

Use Case ID: UC-002

Use Case Name: Check-in Guest

Actor(s): Receptionist

Goal in Context: A guest with a valid reservation is successfully checked into the hotel, assigned a specific room, and their reservation status is updated.

Scope: The Hotel Management System.

Preconditions:

The Receptionist is logged into the system.

The guest has a valid, active reservation for the current date or a past date within a grace period.

There is at least one available room of the reserved type that is clean and ready for occupancy.

Postconditions:

Success: The reservation status is updated to 'Checked-in'. A specific room number is assigned and linked to the reservation record. The assigned room's status is updated to 'Occupied'.

Failure: The reservation status remains unchanged. No room is assigned.

Primary Success Scenario:

The Receptionist initiates the Check-in process for a guest arriving at the hotel, usually through a front desk interface.

The System prompts the Receptionist to search for the guest's reservation using criteria such as guest name, reservation ID, or arrival date.

The Receptionist enters search criteria provided by the guest and submits the search.

The System finds and displays the matching reservation details, including guest name, dates, reserved room type, number of guests, and current status.

The Receptionist confirms the guest's identity and details against the reservation and initiates the check-in action within the system.

The System displays a list of available specific rooms that match the reserved room type and are marked as 'Clean' and 'Available'.

The Receptionist selects a specific room from the list to assign to the guest.
The System assigns the selected room to the reservation record, updates the reservation status to 'Checked-in', and updates the assigned room's status to 'Occupied'.
The System confirms successful check-in to the Receptionist, displaying the assigned room number and updated reservation status.
Alternative Flows:

AF-2.1: Reservation Not Found:

Steps 2-4 (Search reservation): If no reservation is found matching the search criteria, the System informs the Receptionist that the reservation was not found. The Receptionist may re-enter criteria or the Use Case ends.

AF-2.2: Reservation Not Valid for Check-in:

Step 5 (Confirm guest details): If the found reservation is not in a status eligible for check-in (e.g., it's 'Canceled', already 'Checked-in', or the arrival date is in the future beyond a permissible early check-in window), the System informs the Receptionist that the reservation is not eligible for check-in at this time. (End Use Case).

AF-2.3: No Rooms Available of Reserved Type:

Steps 6-7 (Assign room): If there are no clean, available rooms matching the reserved room type at the time of check-in, the System informs the Receptionist. The Receptionist may need to assign a different room type (if allowed by hotel policy, potentially involving a price adjustment handled by billing) or inform the guest there is a delay until a room becomes available and clean. (End Use Case or proceed with a policy-defined alternative flow involving room change/upgrade).

AF-2.4: Guest Arrives Early/Late:

Step 5 (Confirm guest details): If the guest arrives significantly early or late compared to the reservation date, the System may flag this. Policies might require approval (e.g., for early check-in before standard time) or handle late arrivals (e.g., if reservation was marked as a "no-show"). The System informs the Receptionist of any policy implications.

AF-2.5: Guest Requires Different Room:

Steps 6-7 (Assign room): The guest may request a different specific room or room type upon arrival. The System allows the Receptionist to search for and assign an alternative available

room, potentially requiring updates to the reservation's room type and price, triggering related billing requirements.

Use Case 3: Check-out Guest

Use Case ID: UC-003

Use Case Name: Check-out Guest

Actor(s): Receptionist

Goal in Context: A guest's stay is concluded, the final bill is settled, and the reservation and assigned room statuses are updated appropriately.

Scope: The Hotel Management System.

Preconditions:

The Receptionist is logged into the system.

The guest's reservation is currently in 'Checked-in' status.

Postconditions:

Success: The reservation status is updated to 'Checked-out'. The assigned room's status is updated to 'Needs Cleaning'. The final invoice is generated (if not already) and marked as paid.

Failure: The check-out process is not completed. Reservation and room statuses remain unchanged.

Primary Success Scenario:

The Receptionist initiates the Check-out process for a guest who is departing, usually through the front desk interface.

The System prompts the Receptionist to search for the guest's reservation using criteria such as guest name, room number, or reservation ID.

The Receptionist enters search criteria and submits.

The System finds and displays the matching reservation details and associated outstanding charges, presenting or generating the final invoice. This invoice aggregates all charges, including room nights, taxes, and any additional items posted during the stay (e.g., restaurant charges from REQ-3.1.15).

The Receptionist reviews the final invoice details with the guest for confirmation.

The System presents the final invoice detailing the total amount due.
The Receptionist processes the payment for the outstanding balance using a selected payment method (e.g., credit card, cash). This step may involve interacting with the billing and payment features (See Use Case: Process Payment, potentially linked to REQ-3.1.14).
Upon successful payment processing, the System confirms successful payment and marks the invoice as 'Paid' in the system.
The System updates the reservation status to 'Checked-out' and updates the assigned room status to 'Needs Cleaning'.
The System confirms successful check-out to the Receptionist and provides a final receipt or paid invoice to the guest.
Alternative Flows:
AF-3.1: Guest Not Checked In:
Steps 2-4 (Search reservation): If the found reservation is not in 'Checked-in' status, the System informs the Receptionist that the guest is not currently checked in. (End Use Case).
AF-3.2: Payment Failure:
Step 7 (Process Payment): If payment processing fails for the final bill, the System informs the Receptionist of the failure reason and allows them to attempt payment using an alternative method or handles the situation based on hotel policy regarding outstanding balances. (Return to Step 6 or handle via AF-3.3).
AF-3.3: Payment Deferred:
Steps 6-8 (Payment): If the payment is not completed at the time of check-out (e.g., for corporate billing, direct billing, or guest requires invoice later), the System updates the reservation status to 'Checked-out' and room status to 'Needs Cleaning' but leaves the invoice marked with a 'Pending' or 'Balance Due' status. The Post condition is modified accordingly to reflect the pending payment. A separate process or Use Case would handle collecting the deferred payment.
AF-3.4: Outstanding Charges Need Addition:
Steps 4-5 (Display invoice): If there are additional charges incurred by the guest just before check-out that have not yet been posted to the invoice, the Receptionist may need to add these items to the invoice before presenting it. The System allows the addition of items and

recalculates the total. (Involves interaction with related functionalities like POS/Order Management REQ-3.1.15).

AF-3.5: Guest Requests Early Check-out:

Step 5 (Confirm details): If the guest is checking out before their reserved check-out date, hotel policy may apply. The System might calculate the bill differently (e.g., applying an early departure fee or requiring payment for the full reserved stay). The Receptionist confirms the updated charges with the guest.

Use Case 4: Update Room Status

Use Case ID: UC-004

Use Case Name: Update Room Status

Actor(s): Housekeeping Staff, Receptionist, Manager

Goal in Context: The status of a specific physical room is accurately updated and reflected in the system, enabling proper room assignment for new arrivals and managing the housekeeping workflow.

Scope: The Hotel Management System.

Preconditions:

The user (Housekeeping Staff, Receptionist, or Manager) is logged into the system with appropriate permissions.

Rooms are defined and registered in the system.

Postconditions:

Success: The Current Status attribute of the specified Room entity is successfully updated to the new status .

Failure: The room status remains unchanged.

Triggers:

Automatic trigger upon guest check-out (sets status to 'Needs Cleaning').

Housekeeping Staff completes cleaning and reports the room ready.

Receptionist or Manager needs to manually change a room status (e.g., to 'Under Maintenance', 'Available' after inspection).

Primary Success Scenario (Housekeeping Staff - Mark as Clean):

The Housekeeping Staff initiates the process to update room status, typically via a mobile interface or dedicated terminal showing their task list.

The System displays a list of rooms assigned for cleaning or rooms currently marked with 'Needs Cleaning' status. The list includes relevant details like room number and current status.
The Housekeeping Staff selects a specific room from the list that they have finished cleaning and inspected.
The System prompts the Housekeeping Staff to confirm the status update, typically suggesting setting the status to 'Clean' or 'Inspected/Clean'.
The Housekeeping Staff confirms the update action.
The System updates the status of the selected room to 'Clean' in the database.
The System confirms the successful update to the Housekeeping Staff, often by removing the room from their 'Needs Cleaning' list or displaying a success message.
Primary Success Scenario (Receptionist/Manager - Manual Update):
The user (Receptionist or Manager) initiates the process to update room status manually, typically through the main system interface.
The System prompts the user to identify the room whose status needs updating (e.g., by entering the room number or selecting from a list).
The user enters the room identification and submits.
The System finds and displays the current room details, including its current status.
The System presents a list of available statuses that the room can be changed to (e.g., 'Available', 'Occupied', 'Under Maintenance', 'Needs Cleaning', 'Clean', 'Inspected').
The user selects the desired new status from the list.
The System updates the status of the selected room to the chosen status in the database.
The System confirms the successful update to the user, displaying the room number and its new status.
Alternative Flows:
AF-4.1: Room Not Found:
Steps 2-4 (Identify room): If the room cannot be found in the system based on the identification provided, the System informs the user that the room was not found. (End Use Case).

AF-4.2: Invalid Status Selection:

Step 6 (Select new status): If the user attempts to select a status that is logically inconsistent or not permitted based on the room's current state or user's role (e.g., manually setting an 'Occupied' room to 'Clean' without a corresponding check-out process, or a Housekeeping staff trying to mark a room as 'Under Maintenance' if that's restricted to managers), the System displays an error message explaining the restriction. (Return to Step 5 or end Use Case).

AF-4.3: Room Linked to Active Reservation:

Step 7 (Update status): If the user attempts to manually change the status of a room currently assigned to an active 'Checked-in' reservation (e.g., trying to mark an occupied room as 'Needs Cleaning'), the System should prevent this manual update or issue a warning, as it conflicts with the reservation status. Updates like setting a room to 'Under Maintenance' while occupied might require confirmation or specific permissions.

AF-4.4: System Automatically Updates Status:

This is a separate flow triggered by other Use Cases (e.g., Check-out Guest - AF-3.1) or internal system processes (e.g., nightly audits marking no-shows). This isn't initiated by an actor in the same way but represents a system-driven status change, fulfilling REQ-3.1.17. These detailed Use Case descriptions utilize the structured narrative format from the provided sources and cover the main interactions related to reservations and room status, contributing significantly to the documentation length as noted in source.

3.3 Data requirements

This section defines the logical characteristics and structure of the data that the system must manage. It outlines the data that the system will consume, process, and create. Understanding the data requirements is crucial for the subsequent design phase, particularly for developing the database schema. This section focuses on the logical view of the data, independent of specific database management systems.

3.3.1 Overview

The Hotel Management System relies on accurate and well-organized data to perform its functions, such as managing reservations, tracking room status, and processing payments.

This subsection provides an overview of the major data entities within the system, their key attributes, and the relationships between these entities.
This information serves as the foundation for the logical database design and ensures that the system's data model supports all defined functional and non-functional requirements. The representation includes data entities and their relationships.

3.3.2 Entity, Attribute and Relationships

The core data elements of the system are represented as entities. Each entity has associated attributes that describe its properties. Relationships define how entities are connected to each other.

* Entity: Guest
  * Description: Represents an individual customer who interacts with the hotel, potentially making reservations or staying in rooms.
  * Attributes:
    * GuestID: Unique identifier for the guest (e.g., System-generated).
    * FirstName: Guest's first name.
    * LastName: Guest's last name.
    * DateOfBirth: Guest's date of birth (Optional).
    * EmailAddress: Guest's email address (Required, Unique, Valid format).
    * PhoneNumber: Guest's primary phone number (Required, Valid format).
    * Address: Guest's physical address (Street, City, State/Province, Zip/Postal Code, Country).
    * MembershipStatus: (Optional) e.g., 'Standard', 'VIP'.
    * RegistrationDate: Date the guest registered in the system.
  * Relationships:
    * Guest places Reservation (One-to-Many: One Guest can have multiple Reservations over time).
* Entity: Employee
  * Description: Represents hotel staff who use the system (e.g., Receptionist, Housekeeping, Manager).
  * Attributes:

    * EmployeeID: Unique identifier for the employee.
    * FirstName: Employee's first name.
    * LastName: Employee's last name.
    * Username: Unique login username.
    * PasswordHash: Hashed value of the employee's password (Stored securely).
    * Role: Employee's assigned role (e.g., 'Receptionist', 'Housekeeping', 'Manager', 'Admin').
    * ContactInfo: Employee's contact details.
    * HireDate: Date the employee was hired.
    * IsActive: Boolean indicating if the employee account is active.
  * Relationships:
    * Employee manages Reservation (One-to-Many: One Employee can create or modify multiple Reservations).
    * Employee manages Room (One-to-Many: One Employee can update status for multiple Rooms).
    * Employee manages Invoice (One-to-Many: One Employee can process payments for multiple Invoices).
* Entity: RoomType
  * Description: Defines categories of rooms with similar characteristics.
  * Attributes:
    * RoomTypeID: Unique identifier for the room type.
    * TypeName: Name of the room type (e.g., 'Standard', 'Deluxe', 'Suite') (Required, Unique).
    * Description: Detailed description of the room type.
    * DefaultPricePerNight: Standard price for this room type per night.
    * Capacity: Maximum number of guests the room type can accommodate.
    * Amenities: List or description of amenities (e.g., 'Free Wi-Fi', 'Ocean View').
  * Relationships:
    * RoomType has Room (One-to-Many: One Room Type can have multiple physical Rooms).
* Entity: Room

* Description: Represents a single physical room in the hotel.
* Attributes:
    * RoomID: Unique identifier for the room.
    * RoomNumber: The physical room number (Required, Unique).
    * RoomTypeID: Foreign key referencing the associated RoomType.
    * CurrentStatus: Current status of the room (e.g., 'Available', 'Occupied', 'Needs Cleaning', 'Under Maintenance') (Enum/Lookup).
    * LocationDescription: (Optional) e.g., 'Floor 3', 'West Wing'.
* Relationships:
    * Room is of RoomType (Many-to-One: Multiple Rooms belong to one Room Type).
    * Room is assigned to Reservation (One-to-One at any given time: A Room is assigned to at most one active Reservation).

* Entity: Reservation
* Description: Represents a booking made by a guest for a specific period.
* Attributes:
    * ReservationID: Unique identifier for the reservation.
    * GuestID: Foreign key referencing the Guest who made the reservation.
    * ReservedRoomTypeID: Foreign key referencing the RoomType initially reserved.
    * AssignedRoomID: Foreign key referencing the specific Room assigned during check-in (Optional, null before check-in).
    * CheckInDate: The scheduled check-in date (Required).
    * CheckOutDate: The scheduled check-out date (Required).
    * NumberOfGuests: The number of guests included in the reservation (Required).
    * ReservationStatus: Current status of the reservation (e.g., 'Confirmed', 'Checked-in', 'Checked-out', 'Canceled') (Enum/Lookup).
    * BookingDate: The date the reservation was created.
    * EstimatedTotalCost: Calculated cost based on dates and room type price.
* Relationships:
    * Reservation is made by Guest (Many-to-One: Many Reservations can be made by one Guest).

    * Reservation is for RoomType (Many-to-One: Many Reservations are for one Room Type).
    * Reservation is assigned Room (One-to-One: One Reservation can be assigned one Room when checked-in).
    * Reservation generates Invoice (One-to-One: One Reservation has one main Invoice).

* Entity: Invoice
  * Description: Represents the bill issued for a reservation or other services.
  * Attributes:
    * InvoiceID: Unique identifier for the invoice.
    * ReservationID: Foreign key referencing the associated Reservation.
    * InvoiceDate: Date the invoice was generated.
    * DueDate: Date the payment is due (Optional).
    * TotalAmount: The total amount due.
    * AmountPaid: The total amount paid.
    * BalanceDue: Calculated as TotalAmount - AmountPaid.
    * PaymentStatus: Status of the invoice payment (e.g., 'Pending', 'Paid', 'Partially Paid', 'Void') (Enum/Lookup).
    * IssueEmployeeID: Foreign key referencing the Employee who generated/finalized the invoice.
  * Relationships:
    * Invoice is for Reservation (One-to-One: One Invoice is for one Reservation).
    * Invoice includes InvoiceItem (One-to-Many: One Invoice can have multiple line items).
    * Invoice is processed by Employee (Many-to-One: Many Invoices are processed by one Employee).
* Entity: InvoiceItem
  * Description: Represents a single line item on an invoice (e.g., room charge for a night, a restaurant charge).
  * Attributes:
    * InvoiceItemID: Unique identifier for the invoice item.

    * InvoiceID: Foreign key referencing the parent Invoice.
    * Description: Description of the charge (e.g., 'Room Charge - Night 1', 'Restaurant Order').
    * Quantity: Quantity of the item (e.g., number of nights, number of items).
    * UnitPrice: Price per unit.
    * LineTotal: Calculated as Quantity * UnitPrice.
    * ItemDate: Date the charge was incurred.
  * Relationships:
    * InvoiceItem belongs to Invoice (Many-to-One: Many Invoice Items belong to one Invoice).
* Entity: Payment
  * Description: Represents a payment transaction made towards an invoice.
  * Attributes:
    * PaymentID: Unique identifier for the payment.
    * InvoiceID: Foreign key referencing the Invoice being paid.
    * PaymentDate: Date the payment was received.
    * Amount: Amount paid in this transaction.
    * PaymentMethod: Method used for payment (e.g., 'Credit Card', 'Cash', 'Transfer') (Enum/Lookup).
* TransactionDetails: Reference number, card type, last 4 digits, etc. (Sensitive details should be handled securely and potentially not stored directly).
    * ReceivingEmployeeID: Foreign key referencing the Employee who processed the payment.
  * Relationships:
    * Payment applies to Invoice (Many-to-One: Multiple Payments can be applied to one Invoice).
    * Payment is received by Employee (Many-to-One: Many Payments are received by one Employee).
(Note: Entities like InventoryItem, Order, OrderItem could be added if the POS/Restaurant functionality is within the final scope. The above focuses on core hotel operations.)

3.3.3 Entity-Relationship Diagram

The following diagram illustrates the logical relationships between the core entities described above.

Figure 3: Entity-Relationship Diagram

3.3.4 Data Validation Rules

Data validation rules are necessary to ensure the integrity and accuracy of the information stored in the system. These rules define constraints on data entry and manipulation to prevent errors and maintain consistency.

* DR-3.3.4.1: For the Guest entity, the EmailAddress attribute shall be unique across all guest records.
* DR-3.3.4.2: For the Guest entity, the EmailAddress and PhoneNumber attributes shall conform to standard validation formats (e.g., RFC 5322 for email, E.164 for phone numbers, or regionally appropriate equivalents).
* DR-3.3.4.3: For the Room entity, the RoomNumber attribute shall be unique across all room records.
* DR-3.3.4.4: For the Reservation entity, the CheckOutDate shall be after the CheckInDate.
* DR-3.3.4.5: For the Reservation entity, the date range defined by CheckInDate and CheckOutDate for a specific room (once assigned) shall not overlap with the date range of any other reservation assigned to the same room.
* DR-3.3.4.6: For the Reservation entity, the NumberOfGuests shall not exceed the Capacity of the ReservedRoomType.
* DR-3.3.4.7: For the Invoice entity, the TotalAmount shall be the sum of the LineTotal of all associated InvoiceItem records.
* DR-3.3.4.8: For the Invoice entity, the BalanceDue shall be calculated as TotalAmount minus AmountPaid.
* DR-3.3.4.9: For the Payment entity, the Amount shall be a non-negative value.
* DR-3.3.4.10: For the User (Guest/Employee) login, passwords shall meet a minimum complexity requirement (e.g., minimum length, inclusion of uppercase, lowercase, numbers, and symbols).

3.4 External Interface Requirements

3.4.1 Hardware Interface

This subsection describes the logical and physical characteristics of each interface between the software product and the hardware components of the system. For the web-based Hotel Management System, the primary hardware interactions will involve the server infrastructure hosting the application and database, and potentially client-side hardware for users.

* Server Hardware: The system will interface with server hardware responsible for processing requests, storing data, and running the application code. Requirements related to processing power, memory (RAM), storage capacity, and network connectivity of the server hardware should be considered here. The system must be able to operate within specified hardware resource limits.
  * REQ-3.4.1.1: The system shall be deployable on server hardware meeting minimum specifications for CPU, RAM, and storage to support expected user load and data volume.
    * Details: Specific CPU type/speed, minimum RAM (e.g., based on expected number of concurrent users), minimum storage for database and application files.
* Database Server Hardware: The database component will require dedicated or shared server hardware. This hardware must provide sufficient I/O speed and storage capacity to handle database transactions efficiently.
  * REQ-3.4.1.2: The database component shall be supported by server hardware providing adequate disk I/O performance for database read/write operations.
    * Details: Specify requirements for storage type (e.g., SSD recommended) and disk throughput.
* Client Hardware: Users (Administration, Staff, Authorized users) will access the system via web browsers on various devices such as desktop computers, laptops, or tablets. The software interfaces with this hardware indirectly through the web browser, which handles rendering and input/output via standard peripherals.

* REQ-3.4.1.3: The user interface shall be compatible with standard input devices (keyboard, mouse, touch screens) and display devices (monitors, tablet screens) on common computing hardware accessing the web browser.
    * Details: Ensure compatibility with standard resolutions and input methods.
* Printing Devices: The system may need to interface with printing hardware for generating reports, invoices, or reservation confirmations. This interface is typically managed through the operating system and web browser printing capabilities.
* REQ-3.4.1.4: The system shall support printing of reports, invoices, and confirmations via standard web browser printing functionality to connected or network printers.
* Point of Sale (POS) Hardware: If the system includes Point of Sale functionality for the restaurant or other services, it might need to interface with specific POS hardware like receipt printers, cash drawers, bar code scanners, or payment terminals. The nature of data and control interactions for these devices needs to be defined.
* REQ-3.4.1.5: If POS functionality is implemented, the system shall interface with standard POS receipt printers to generate guest receipts for orders.
    * Details: Specify required printer connectivity types (e.g., USB, Ethernet) and printing protocols if necessary.
* REQ-3.4.1.6: If integrated payment processing is implemented via POS, the system shall interface with supported payment terminals for processing credit/debit card transactions.
    * Details: Describe the communication method (e.g., direct connection, network) and protocols used.
* Network Hardware: The system relies on network infrastructure (routers, switches, firewalls) for communication between the server, database, and client devices.
* REQ-3.4.1.7: The system shall function correctly over standard network infrastructure with typical latency and bandwidth characteristics for web applications.
    * Details: Define minimal network performance requirements if critical.

3.4.2 Software Interface

This subsection describes the logical and physical characteristics of each interface between the software product and other software components or systems. This includes operating systems, database management systems, middleware, and applications the system interacts with.

* Operating System: The server-side software will run on a specific operating system. Client-side access is via web browsers running on various operating systems.
  * REQ-3.4.2.1: The server-side application shall be compatible with a specified server operating system (e.g., Linux, Windows Server).
    * Details: List the supported operating system(s) and version(s).
  * REQ-3.4.2.2: The user interface shall be accessible via a web browser running on common client operating systems (e.g., Windows, macOS, Linux, iOS, Android).
* Database Management System (DBMS): The system requires an interface to a DBMS to store and manage persistent data. While the SRS should avoid requiring a *specific* software package if possible, it must define the interface requirements.
  * REQ-3.4.2.3: The application shall interface with a relational database management system using standard database connectivity protocols (e.g., JDBC, ODBC, or database-specific connectors).
    * Details: Specify the required database features (e.g., support for SQL, transactions, referential integrity).
  * REQ-3.4.2.4: The system shall handle database interactions through defined APIs or connection methods to ensure data integrity and security.
* Web Server Software: The application will interface with web server software to handle HTTP requests and serve web content.
  * REQ-3.4.2.5: The application shall be compatible with standard web server software (e.g., Apache, Nginx, IIS).
    * Details: List supported web servers and versions.
* Email System: The system may need to send email notifications for reservations, confirmations, or other events. This requires an interface to an email server or service.

* REQ-3.4.2.6: The system shall interface with an external email service or server to send automated email notifications to users.

    * Details: Specify the required protocol (e.g., SMTP) and necessary configuration parameters.

* Payment Gateway Software: If online payment processing is included, the system will interface with a payment gateway's API or software library.

* REQ-3.4.2.7: REQ-3.4.2.7: The system shall integrate with Chapa (https://chapa.co), an Ethiopian online payment gateway, using its RESTful API. It shall support transaction initiation, payment confirmation, and webhook handling for status updates.

* REQ-3.4.2.8: The system shall interface with necessary third-party libraries for functionalities such as front-end frameworks (e.g., React mentioned in proposal), back-end frameworks (e.g., Node.js/PHP mentioned in proposal), or data processing.

    * Details: List specific libraries or types of libraries and their roles.

3.4.3 User Interface

This subsection describes the logical characteristics of each interface between the software product and the users. This includes describing the interface for each defined user class. Detailed design aspects should be documented separately. The interface should follow defined standards or style guides.

User Classes: The system will primarily interact with three user classes: Administration, Staff (e.g., Receptionist, Restaurant Staff), and Authorized users (Customers). Each class will have different interface needs based on their roles and permissions.

* REQ-3.4.3.1: The system shall provide distinct user interfaces tailored to the roles and permissions of Administration, Staff, and Authorized User classes.

* General Interface Characteristics: The user interface should be intuitive and user-friendly.

* REQ-3.4.3.2: The user interface shall provide a consistent navigation structure across different sections of the system.

* REQ-3.4.3.3: The system shall provide clear and informative feedback to the user for all actions, including confirmations, errors, and progress updates.

* Web Interface: As a web-based system, the primary interface will be accessed via standard web browsers.

* REQ-3.4.3.4: The user interface shall be accessible via recent versions of major web browsers (e.g., Chrome, Firefox, Edge, Safari).

* REQ-3.4.3.5: The web interface shall be responsive and adapt to different screen sizes (desktop, tablet, mobile) to provide a usable experience across devices.

* Input and Output: Define the format and organization for data entry screens, forms, reports, and messages.

* REQ-3.4.3.6: Input forms shall include clear labels, required field indicators, and appropriate input controls (e.g., text boxes, dropdowns, date pickers).

* REQ-3.4.3.7: Output data, such as search results, reports, and guest details, shall be presented in a clear, organized, and readable format.

* REQ-3.4.3.8: The system shall provide error messages that are user-friendly, clearly indicate the problem, and suggest potential solutions.

* Specific User Interface Needs (by User Class):

    * *Administration Interface:* Requires interfaces for system configuration, user management, data maintenance (e.g., room types, rates), report generation (e.g., sales reports), and potentially inventory management if centralized.

    * *Staff Interface (Receptionist):* Requires interfaces for managing reservations (booking, check-in/check-out), room assignments, guest management, processing payments, and accessing guest information.

    * *Staff Interface (Restaurant/POS):* Requires interfaces for taking orders (online and potentially in-person), managing tables/orders, processing payments, and managing restaurant inventory.

    * *Authorized User Interface (Customer):* Requires interfaces for searching for rooms/tables, viewing availability, making/canceling reservations, online ordering, managing their profile, and viewing order/reservation history.

* REQ-3.4.3.9: The Staff interface shall provide a clear view of room availability and reservation details to facilitate the check-in process.

* REQ-3.4.3.10: The Authorized User interface shall allow users to view the restaurant menu and place online orders.

3.4.4 Communication Interface

This subsection describes the logical and physical characteristics of the interfaces to any communications functions, such as networks, communication protocols, and message formats.

* Network Protocols: The system relies on standard internet protocols for communication.
* REQ-3.4.4.1: The system shall use HTTP/HTTPS protocols for communication between the user's web browser and the server.
* REQ-3.4.4.2: Secure communication (HTTPS) shall be used for all user logins, data transmission containing sensitive information (e.g., personal details, payment information), and critical administrative functions.

* Database Communication: Communication between the application server and the database server uses specific protocols depending on the DBMS.
* REQ-3.4.4.3: Communication with the database server shall utilize the native protocol of the chosen DBMS or a standard protocol like TCP/IP with the appropriate database driver.
* Email Communication: Communication with the email server will use email protocols.
* REQ-3.4.4.4: The system shall communicate with the email server using the SMTP protocol.
* Payment Gateway Communication: Integration with a payment gateway involves specific communication protocols or API standards.
* REQ-3.4.4.5: Communication with the payment gateway shall adhere to the specific API specifications and security protocols required by the chosen provider. This typically involves secure data transmission (e.g., TLS/SSL).
* Data Formats: Define the format of data exchanged over these interfaces, such as XML, JSON, or specific message structures.

* REQ-3.4.4.6: Data exchanged between the application server and the web browser shall primarily use standard web data formats such as HTML, CSS, JSON, and potentially XML for specific data transfers.

3.5 Non-functional Requirements

3.5.1 Performance Requirements

Performance requirements define acceptable response times, throughput, capacity, and other quantitative measures of system behavior under various conditions. The performance of the system will depend on hardware and software components.

* Response Time: The time taken for the system to respond to a user action.
* NFR-3.5.1.1: The load time for user interface screens (excluding initial application load) shall take no longer than two seconds under normal load conditions.
* NFR-3.5.1.2: User login authentication shall be verified within five seconds.
* NFR-3.5.1.3: Search queries for rooms or menu items shall return results within five seconds under normal database load.
* NFR-3.5.1.4: Processing a new reservation request shall complete within three seconds, including updating database records and initiating email notifications.
* NFR-3.5.1.5: Generating daily or weekly reports (e.g., sales report) shall not take longer than 30 seconds during off-peak hours.
* Throughput: The number of transactions or operations the system can process within a given time period.
* NFR-3.5.1.6: The system shall support a minimum of 50 concurrent active user sessions without significant degradation in response time.
* NFR-3.5.1.7: The system shall be capable of processing 100 reservation requests per hour during peak periods.

* Capacity: The maximum amount of data the system can store or the maximum number of users it can handle.

* NFR-3.5.1.8: The system shall be capable of storing reservation and transaction data for at least 2 year

* NFR-3.5.1.9: The database shall accommodate data for up to 50 rooms and 100 customer records.

* Scalability: The ability of the system to handle increasing load or data volume by adding resources.

* NFR-3.5.1.10: The system architecture should be designed to allow for horizontal scaling of application servers and database capacity to handle future growth in user base and data volume.

3.5.2 Usability Requirements

Usability requirements define how easy the system is to understand, learn, operate, and use effectively. This relates to user satisfaction and efficiency. The system should provide user-friendly functions and attractive interfaces.

* NFR-3.5.2.1: The user interface shall be intuitive and easy for staff members (Receptionist, Restaurant Staff) with basic computer literacy to learn and operate with minimal training.

* NFR-3.5.2.2: Authorized users (customers) shall be able to navigate the public web interface, search for rooms/tables, and complete reservations or online orders without requiring instructions or assistance.

* NFR-3.5.2.3: The system shall minimize the number of steps required to complete common tasks, such as checking a guest in or placing a restaurant order.

* NFR-3.5.2.4: User input errors shall be clearly indicated, and the system shall provide guidance on how to correct them.

* NFR-3.5.2.5: Help or documentation (User Manuals) shall be available to assist users with system functions.
* Details: A hard copy of the user manual will be delivered, written in simple language.

3.5.3 Security Requirements

Security requirements specify the measures taken to protect the system and data from unauthorized access, use, disclosure, disruption, modification, or destruction. Given the sensitive nature of customer and financial data, security is critical.

* Authentication: Verifying the identity of a user or system process.
* NFR-3.5.3.1: All users (Administration, Staff, Authorized Users) shall be required to authenticate with a unique username and password to access restricted areas of the system.
* NFR-3.5.3.2: Password policies shall be enforced, requiring minimum length and complexity.
* NFR-3.5.3.3: The system shall implement mechanisms to prevent brute-force login attempts (e.g., account lockout after multiple failed attempts).
* Authorization: Defining and enforcing access controls based on user roles and permissions.
* NFR-3.5.3.4: Access to sensitive functions (e.g., deleting records, modifying user roles, viewing detailed financial reports) shall be restricted to authorized Administration or management staff only.
* NFR-3.5.3.5: Staff users shall only have access to functions and data necessary for their specific roles (e.g., Receptionist access to reservations, Restaurant Staff access to orders).
* NFR-3.5.3.6: Authorized users (customers) shall only have access to their own profile information, reservation history, and online ordering functions.
* Data Protection and Privacy: Protecting sensitive data at rest and in transit.
* NFR-3.5.3.7: All transmission of sensitive data (e.g., personal information, payment details, passwords) between the user's browser and the server shall be encrypted using industry-standard protocols (e.g., TLS/SSL).

* NFR-3.5.3.8: Customer personal information and payment details stored in the database shall be protected using appropriate security measures (e.g., encryption for sensitive fields, access controls).

* NFR-3.5.3.9: The system shall comply with relevant data privacy regulations (e.g., GDPR, if applicable based on location of users/data). While not explicitly stated in sources, this is a common legal requirement for systems handling personal data, and aligns with data integrity requirements.

* Auditing and Logging: Recording security-relevant events.

* NFR-3.5.3.10: The system shall log all successful and failed login attempts, as well as critical administrative actions.

* NFR-3.5.3.11: System logs shall be protected from unauthorized access or modification.

* Protection Against Common Threats: Protecting against vulnerabilities specific to web applications.

* NFR-3.5.3.12: The system shall implement measures to prevent common web security vulnerabilities, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

3.5.4 Software Quality Attributes

This subsection describes other quality characteristics of the software system. These attributes contribute to the overall quality and success of the system.

* Reliability: The extent to which the program performs with required precision and consistency over time, especially under specified conditions. It can be specified using measures like Mean Time Between Failures (MTBF) or Mean Time to Recovery (MTTR).

* NFR-3.5.4.1: The system shall have a reliability of more than 80% uptime during normal hotel operating hours.

* NFR-3.5.4.2: The system shall correctly process all valid reservation, order, and payment transactions without data corruption or loss.

* NFR-3.5.4.3: The system shall be able to recover from unexpected failures (e.g., power outage, server crash) with a Mean Time to Recovery (MTTR) of no more than 30 minutes

* Availability: The degree to which a system or component is operational and accessible when needed.

* NFR-3.5.4.4: The system shall be available during normal hotel operating hours. For the web interface, this might extend to 24/7 availability for customer access (online booking/ordering). Specify exact required hours.

* Maintainability: The effort required to modify the software to correct defects, improve performance or other attributes, or adapt to a changed environment. This includes ease of fixing bugs and implementing updates.

* NFR-3.5.4.5: The system code base shall be structured and documented to allow a different development team to understand, modify, and fix defects efficiently. Adherence to coding standards is important.

* NFR-3.5.4.6: Implementing minor functional updates (e.g., adding a new room type, updating a menu item) should be achievable with minimal effort and risk.

* Portability: The ease with which the software can be transferred from one hardware or software environment to another. For a web application, this relates to the server environment and client browser compatibility.

* NFR-3.5.4.7: The server-side application should be designed to be portable across different operating system distributions within the specified family (e.g., different Linux distributions).

* NFR-3.5.4.8: The user interface shall be portable across different major web browsers and operating systems on client devices.

* Robustness: The strength of the system to handle errors and unexpected situations gracefully and maintain database integrity without facing unexpected failures.

* NFR-3.5.4.9: The system shall handle invalid user input, network interruptions, and temporary unavailability of external services (e.g., email server) without crashing or losing data.

* NFR-3.5.4.10: Critical operations, such as processing payments or confirming reservations, shall be atomic and designed to maintain data consistency even in the event of system interruption (e.g., using database transactions).

* Efficiency: The amount of computing resources (e.g., CPU, memory, disk space, network bandwidth) and time required for the system to perform its functions. Performance requirements often specify aspects of efficiency.

* NFR-3.5.4.11: The system shall manage memory and processing resources efficiently to minimize operational costs.

* NFR-3.5.4.12: Database queries should be optimized to minimize load on the database server.

* Integrity: How the system secures information and avoids data losses, including referential integrity in database tables. This is closely related to security and reliability.

* NFR-3.5.4.13: The system shall enforce referential integrity constraints in the database to ensure consistent relationships between data entities (e.g., a booking must be linked to a valid guest and room).

* NFR-3.5.4.14: The system shall implement data validation rules upon input to ensure data accuracy and adherence to defined business rules.

* Flexibility: The ability to add new features to the system and handle them conveniently.

* NFR-3.5.4.15: The system architecture should be designed to facilitate the addition of new features (e.g., loyalty programs, event booking) in future iterations with minimal impact on existing functionality.

* Reusability: The extent to which components of the system can be reused in other applications.

* NFR-3.5.4.16: System components, such as the authentication module or database access layer, should be developed with potential for reuse in other projects where feasible. This is typically a 'should' requirement as extensive re-usability often involves additional effort.

* Test-ability: The effort needed to test the system to ensure it performs as intended.

* NFR-3.5.4.17: The system shall be designed to allow for efficient testing of individual components and integrated functionality. This might involve providing interfaces or hooks for automated testing.