# System Design Document

# 1. Introduction

# 2. Design Goals

# Design patterns

# 3. Current System Architecture

# 4. Proposed System Architecture

# Architectural style

## 4.1. Architectural Design

## System Architecture

### 4.1.1 System Decomposition

### 4.1.2 Component Modeling

### 4.1.3. Deployment Modeling

## 4.2 Detail Design

### 4.2.1 Class Model

#### 4.2.1.1 Class Description

### 4.2.2 Access control and Security

### 4.2.3 Data Base/Persistence Model

### 4.2.4. Algorithm/Logic Design

## 4.3. User Interface Design

## 4.4. Implementation

# 5. References

# INTRODUCTION

The hotel management system is a comprehensive software solution designed to streamline and automate the operations and processes within a hotel environment. This system aims to simplify various aspects of hotel management, including guest reservations, check-ins and check-outs, room assignments, billing and invoicing, inventory management, and reporting. By leveraging technology and automation, the hotel management system enables efficient and effective management of hotel resources, enhances guest experiences, and optimizes operational workflows.

The purpose of this document is to provide a detailed software architecture design for the hotel management system, outlining its key components, interactions, and functionalities. This document serves as a blueprint for the development team, guiding the implementation phase and ensuring a coherent and scalable system design.

The hotel management system is intended to benefit multiple stakeholders, including hotel staff, guests, and administrators. With a user-friendly interface and intuitive features, hotel staff can easily manage reservations, monitor room availability, process payments, and generate reports. Guests will experience a seamless booking process, personalized services, and a smooth check-in and check-out experience. Administrators will have access to comprehensive reports and analytics to monitor hotel performance, make informed decisions, and enhance overall operational efficiency.

Throughout this document, we will present the proposed system architecture, including the architectural style, system decomposition, component modeling, and deployment strategy. Additionally, we will delve into the detail design aspects, such as the class model, access control and security measures, database/persistence model, user interface design, and implementation considerations.

By adopting this hotel management system, hotels can improve operational efficiency, deliver exceptional guest experiences, and optimize resource utilization. This document serves as a comprehensive guide to the design and implementation of the hotel management system, ensuring a robust and scalable solution tailored to the specific needs of the hotel industry.

# 2. Design goals

The design goals of Hotel management system are:

1. Efficient Reservation Management:
   * Design a system that allows efficient and streamlined management of room reservations.
   * Enable easy booking, modification, and cancellation of reservations.
   * Provide real-time availability updates to avoid double bookings.
2. Seamless Guest Experience:
   * Design a user-friendly interface that enhances the guest experience.
   * Enable guests to easily search for rooms, view amenities, and make reservations.
   * Provide personalized services, such as room preferences and special requests.
3. Effective Inventory Management:
   * Design a system that efficiently tracks and manages hotel room inventory.
   * Enable automated updates of room availability based on bookings, check-ins, and check-outs.
   * Implement alerts for low inventory levels or maintenance requirements.
4. Comprehensive Billing and Payment System:
   * Design a robust billing and payment system for accurate and timely invoicing.
   * Integrate with payment gateways to facilitate secure online payments.
   * Generate detailed bills that include room charges, additional services, and taxes.
5. Efficient Staff Management:
   * Design features that support the management of hotel staff and their activities.
   * Provide role-based access control to ensure appropriate access to system functionalities.
   * Enable staff scheduling, task assignment, and performance tracking.
6. Effective Reporting and Analytics:
   * Design a system that generates comprehensive reports and analytics.
   * Provide insights into room occupancy, revenue, guest preferences, and other key metrics.
   * Enable data-driven decision-making for hotel management.
7. Integration with Third-Party Systems:
   * Design the system to integrate seamlessly with external systems and services.
   * Examples include integration with online travel agencies (OTAs), point-of-sale (POS) systems, or accounting software.
   * Ensure smooth data exchange and interoperability between systems.
8. Data Security and Privacy:
   * Design the system with strong security measures to protect guest and hotel data.
   * Implement encryption, access controls, and regular data backups.
   * Comply with data privacy regulations and industry best practices.
9. Scalability and Performance:
   * Design the system to handle a growing number of hotels, rooms, and users.
   * Ensure high performance even during peak times.
   * Implement scalability strategies, such as load balancing and caching.
10. Customization and Flexibility:
    * Design the system to accommodate the specific needs and branding of different hotels.
    * Allow for customization of workflows, branding elements, and business rules.
    * Provide configuration options to adapt to various hotel management scenarios.

# 3 Design patterns

Here is the design pattern used for the Hotel management system.

1. Factory Pattern:
   * The Factory Pattern is chosen to address the need for flexible and consistent object creation within the hotel management system.
   * By encapsulating the object creation logic, the Factory Pattern allows for the easy management of different types of hotel rooms, services, or other objects.
   * It promotes code reusability and maintainability by centralizing the creation logic, making it easier to add new types of objects without modifying the client code.
   * The Factory Pattern provides a clear separation between the client code and the object creation process, improving code organization and modularity.
2. Facade Pattern:
   * The Facade Pattern is selected to simplify the complexity of interacting with the hotel management system, providing a unified and simplified interface.
   * By acting as a higher-level abstraction, the Facade Pattern offers a well-defined and consistent API for common operations.
   * It shields the client code from the complexities of the underlying subsystems or components, making it easier to use and understand.
   * The Facade Pattern enhances code readability, maintainability, and ease of use, especially for developers who may not be familiar with the inner workings of the system.

By using the Factory Pattern and Facade Pattern together, our hotel management system benefits from:

* Modular and flexible object creation: The Factory Pattern allows for the creation of different types of hotel objects in a consistent and flexible manner, facilitating the management of complex entities within the system.
* Simplified and unified interface: The Facade Pattern provides a streamlined and easy-to-use interface, shielding the client code from the complexities of the underlying subsystems and making it more intuitive to interact with the system.

# Current System Architecture

Before the design of a hotel management system, hotels typically have a traditional architecture that encompasses various physical and manual processes.

1. Front Desk:
   * The front desk serves as the central point of interaction between guests and hotel staff.
   * Staff members handle check-ins, check-outs, reservations, and inquiries manually using paper records or basic computer systems.
   * Physical registers or logbooks are used to record guest information, room assignments, and billing details.
2. Reservation System:
   * Hotels may have a basic reservation system, often manual or spreadsheet-based, to manage room availability and bookings.
   * Reservations are recorded and tracked manually, and communication with guests is typically done through phone calls or emails.
3. Room Assignment and Housekeeping:
   * The room assignment process is often done manually, with staff physically assigning rooms to guests based on availability and preferences.
   * Housekeeping activities, such as cleaning and maintenance schedules, are coordinated through manual communication between housekeeping staff and front desk personnel.
4. Billing and Payment:
   * Billing processes are typically manual, involving the creation of handwritten or printed invoices that are given to guests upon check-out.
   * Payment methods vary, including cash, credit cards, or traveler's checks, and are processed manually through terminals or cash registers.
5. Reporting and Analytics:
   * Reporting and analytics in a non-automated environment rely heavily on manual data collection and analysis.
   * Staff manually compiles reports on room occupancy, revenue, expenses, and other key metrics using data from various sources.
6. Communication and Collaboration:
   * Communication between hotel staff members and departments typically occurs through in-person interactions, phone calls, or paper-based messaging systems.
   * Collaboration and coordination among different departments (e.g., front desk, housekeeping, maintenance) are primarily done manually through physical handoffs or meetings.
7. Guest Services:
   * Guest services, such as concierge assistance, room service, or transportation arrangements, are managed through manual processes and direct guest interactions with hotel staff.

# Proposed architectural design

The proposed design architecture for a hotel management system typically aims to automate and streamline various processes involved in hotel operations. Here are key components and characteristics of the proposed design architecture:

1. Centralized Database:
   * The system would feature a centralized database to store and manage essential data such as guest information, room inventory, reservations, billing details, and historical records.
   * The database would provide a reliable and scalable solution for data storage, ensuring data integrity and accessibility.
2. Reservation and Booking System:
   * A robust reservation and booking system would be implemented to enable guests to make reservations online or through various channels.
   * The system would allow real-time availability checks, room selection, and booking confirmations, providing a seamless and convenient experience for guests.
3. Room Management:
   * The system would facilitate efficient room management, including automated room assignment based on guest preferences, availability, and specific requirements.
   * It would track room statuses (occupied, vacant, under cleaning, etc.) and manage housekeeping schedules, ensuring timely cleaning and maintenance of rooms.
4. Guest Services and Communication:
   * The architecture would incorporate features to enhance guest services and communication.
   * This could include self-service kiosks for check-ins and check-outs, automated room service requests, and communication channels for guests to interact with hotel staff and request assistance.
5. Billing and Payment System:
   * A comprehensive billing and payment system would be integrated into the architecture, automating the process of generating invoices and handling various payment methods.
   * This system could include features such as online payment gateways, integration with POS systems, and automated invoicing.
6. Reporting and Analytics:
   * The proposed architecture would include robust reporting and analytics capabilities.
   * It would generate real-time reports on room occupancy rates, revenue, guest preferences, and other key performance indicators to support data-driven decision-making.
7. Integration with External Systems:
   * The architecture would support integration with external systems and services.
   * This could include interfaces with online travel agencies (OTAs) for seamless reservation synchronization, integration with accounting software for financial management, or integration with keycard systems for room access control.
8. Security and Access Control:
   * The proposed design architecture would prioritize security and access control measures to safeguard sensitive data and ensure system integrity.
   * This could include user authentication, role-based access controls, encryption of sensitive information, and adherence to industry-standard security practices.
9. Scalability and High Availability:
   * The architecture would be designed to accommodate scalability and high availability.
   * This could involve load balancing techniques, redundant infrastructure, and cloud-based solutions to ensure optimal performance during peak demand periods and minimize downtime.

# Architectural style

## 6. Architectural Design

## 6.1 System decomposition

System decomposition in a hotel management system involves identifying and breaking down the system into logical components or subsystems based on their responsibilities and functionalities. Here are some potential subsystems parts of the decomposition:

1. Reservation Subsystem:
   * Handles the management of room reservations, including availability checking, booking, modification, and cancellation.
   * Manages guest information, room preferences, and booking history.
2. Room Management Subsystem:
   * Deals with tasks related to room inventory and allocation.
   * Tracks room availability, status (cleaning, maintenance, etc.), and assigns rooms to guests based on their preferences and booking information.
3. Guest Services Subsystem:
   * Handles guest-related services such as check-in, check-out, and concierge services.
   * Manages guest requests, room service orders, and facilitates communication between guests and hotel staff.
4. Billing and Payment Subsystem:
   * Responsible for generating invoices, managing billing details, and handling different payment methods.
   * Integrates with external payment gateways, accounting systems, or POS (Point of Sale) systems to process payments.
5. Reporting and Analytics Subsystem:
   * Collects and analyzes data to generate reports on room occupancy, revenue, guest preferences, and other key performance indicators.
   * Provides insights and analytics to support decision-making processes and help in monitoring the hotel's performance.
6. User Management Subsystem:
   * Handles user authentication, authorization, and management of different user roles and permissions within the system.
   * Manages access control, user profiles, and security measures to safeguard sensitive information.
7. Integration Subsystem:
   * Facilitates integration with external systems, APIs, or third-party services.
   * Enables seamless data exchange with online travel agencies (OTAs), property management systems, or other systems used by the hotel.
8. Administration and Configuration Subsystem:
   * Provides tools and interfaces for system administrators to configure and manage various aspects of the hotel management system.
   * Allows customization of system settings, user roles, rates, taxes, and other administrative functions.

## 6.2 Component Modeling

System component modeling involves defining the different components of the hotel management system and specifying their roles, interfaces, methods, and data structures. Here are some components that can be considered for modeling:

1. Reservation Component:
   * Responsible for handling the reservation-related functionality, including availability checking, booking, modification, and cancellation.
   * Interfaces: ReservationService, ReservationRepository
   * Methods: checkAvailability(), createReservation(), modifyReservation(), cancelReservation()
2. Room Management Component:
   * Manages room inventory, allocation, and status tracking.
   * Interfaces: RoomManagementService, RoomRepository
   * Methods: getAvailableRooms(), assignRoom(), updateRoomStatus()
3. Guest Services Component:
   * Handles guest-related services, such as check-in, check-out, and concierge services.
   * Interfaces: GuestService, ConciergeService
   * Methods: checkInGuest(), checkOutGuest(), handleConciergeRequest()
4. Billing and Payment Component:
   * Deals with billing generation, payment processing, and financial management.
   * Interfaces: BillingService, PaymentGateway
   * Methods: generateInvoice(), processPayment(), calculateRevenue()
5. Reporting and Analytics Component:
   * Provides reporting and analytics capabilities for monitoring system performance and generating insights.
   * Interfaces: ReportingService, AnalyticsService
   * Methods: generateOccupancyReport(), generateRevenueReport(), performDataAnalysis()
6. User Management Component:
   * Manages user authentication, authorization, and role-based access control within the system.
   * Interfaces: UserService, AuthenticationProvider
   * Methods: authenticateUser(), authorizeUser(), createUser()
7. Integration Component:
   * Handles integration with external systems, APIs, or third-party services.
   * Interfaces: IntegrationService, ExternalAPI
   * Methods: synchronizeReservations(), communicateWithPaymentGateway()
8. Administration and Configuration Component:
   * Allows system administrators to configure and manage various aspects of the hotel management system.
   * Interfaces: AdminService, ConfigurationService
   * Methods: updateSystemSettings(), manageUserRoles()

Top of Form

Bottom of Form

## 6.3. Deployment Modeling

Client Workstation" typically represents the hardware or device used by clients or end-users to access and interacts with the hotel management system. Here are some possible elements you could include under the "Client Workstation" in your deployment diagram:

1. Client Application:
   * Represents the software component or application installed on the client workstation that allows users to access the hotel management system.
   * This could be a desktop application, a web-based application accessed through a web browser, or a mobile application.
2. User Interface Components:
   * Represents the graphical user interface (GUI) elements and components that users interact with on the client workstation.
   * This may include forms, screens, buttons, menus, and other user interface controls.
3. Communication Channels:
   * Represents the communication channels used by the client workstation to connect and communicate with the application server.
   * This could include network connections, protocols (such as HTTP or TCP/IP), or other communication mechanisms.
4. Hardware Components:
   * Represents the hardware elements of the client workstation, such as the computer or mobile device.
   * This could include the device itself, the operating system running on it, and any specific hardware requirements or dependencies.
5. Dependencies:
   * Represents any software dependencies or libraries required by the client application to run on the client workstation.
   * This could include libraries, frameworks, or other software components needed for the proper functioning of the client application.

# 6.4 Detail Design

## 6.4.1 Class description

This is the description of each class and their methods in the hotel management system:

1. Address:
   * Description: Represents the address information for a location.
   * Attributes: street, city, state, zip Code, country.
   * Methods: Getter and setter methods for the attributes.
2. Account:
   * Description: Represents an account for a user in the system.
   * Attributes: id, password, status.
   * Methods: Getter and setter methods for the attributes, resetPassword().
3. Person:
   * Description: Represents a person in the system.
   * Attributes: name, address, email, phone, account.
   * Methods: Getter and setter methods for the attributes.
4. Guest (extends Person):
   * Description: Represents a guest in the hotel management system.
   * Attributes: totalRoomsCheckedIn.
   * Methods: Getter and setter methods for the attributes, getBookings().
5. Receptionist (extends Person):
   * Description: Represents a receptionist in the hotel management system.
   * Methods: Getter and setter methods for the attributes, searchMember(), createBooking().
6. Server (extends Person):
   * Description: Represents a server in the hotel management system.
   * Methods: Getter and setter methods for the attributes, addRoomCharge().
7. HotelLocation:
   * Description: Represents a location of a hotel.
   * Attributes: name, address.
   * Methods: Getter and setter methods for the attributes, getRooms().
8. Hotel:
   * Description: Represents a hotel in the system.
   * Attributes: name, locations.
   * Methods: Getter and setter methods for the attributes, addLocation().
9. Room:
   * Description: Represents a room in a hotel.
   * Attributes: roomNumber, style, status, bookingPrice, isSmoking, keys, houseKeepingLog.
   * Methods: Getter and setter methods for the attributes, isRoomAvailable(), checkIn(), checkOut(), search().
10. RoomKey:
    * Description: Represents a key for a room.
    * Attributes: keyId, barcode, isActive, isMaster.
    * Methods: Getter and setter methods for the attributes, assignRoom(), isActive().
11. RoomBooking:
    * Description: Represents a booking of a room.
    * Attributes: reservationNumber, startDate, durationInDays, status, checkIn, checkOut, guestId, room, invoice, notifications.
    * Methods: Getter and setter methods for the attributes, fetchDetails().
12. RoomCharge:
    * Description: Represents a charge associated with a room.
    * Attributes: issuedAt.
    * Methods: Getter and setter methods for the attributes, addInvoiceItem().
13. Amenity (extends RoomCharge):
    * Description: Represents an additional amenity charge for a room.
    * Attributes: name, description.
14. RoomService (extends RoomCharge):
    * Description: Represents a room service charge for a room.
    * Attributes: isChargeable, requestTime.
15. KitchenService (extends RoomCharge):
    * Description: Represents a kitchen service charge for a room.
    * Attributes: description.
16. Invoice:
    * Description: Represents an invoice for a booking.
    * Attributes: invoiceNumber, generatedAt, items.
    * Methods: Getter and setter methods for the attributes, addItem()

**17 HmsMain** class:

* + Contains the main method as the entry point for the program execution.

**18.BookingStatus** enum:

* + Represents different booking statuses.
  + Constants: ACTIVE, CANCELLED, COMPLETED, PENDING.

**19.RoomStatus** enum:

* + Represents different room statuses.
  + Constants: AVAILABLE, OCCUPIED, MAINTENANCE, RESERVED.

**20.RoomStyle** enum:

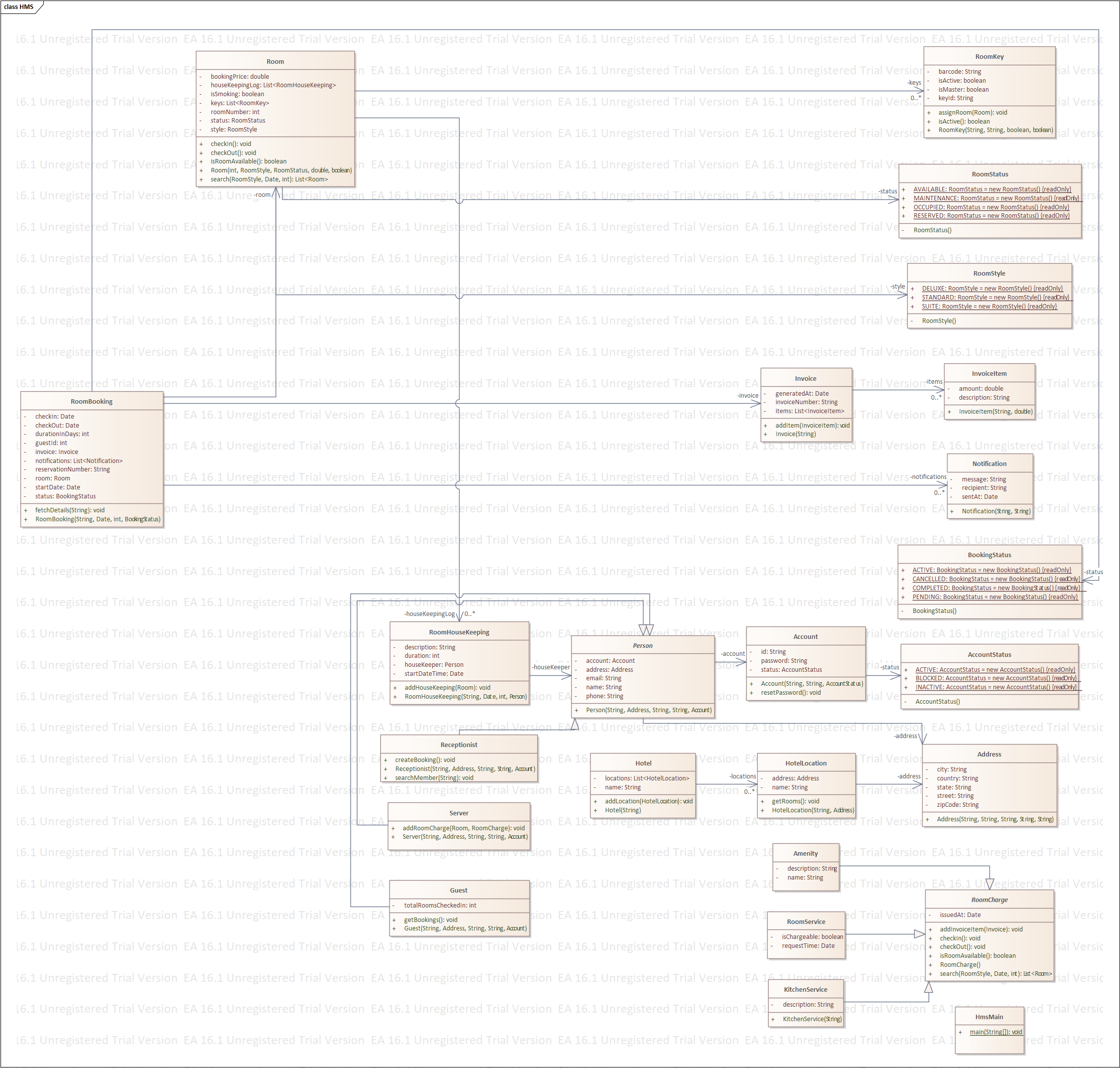
* + Represents different room styles.
  + Constants: STANDARD, DELUXE, SUITE.

**21.AccountStatus** enum:

* + Represents different account statuses.

These classes and their methods provide the foundation for managing hotel-related entities and operations in the hotel management system.

## 6.4.2Class Diagram



## .6.5 Database design

### 6.5.1 Database table

This is the tables of the database for hotel management system

1. **Table: Address**
   * Columns:
     + address\_id (Primary Key)
     + street
     + city
     + state
     + zip\_code
     + country
2. **Table: Account**
   * Columns:
     + account\_id (Primary Key)
     + id
     + password
     + status
3. **Table: Person**
   * Columns:
     + person\_id (Primary Key)
     + name
     + address\_id (Foreign Key)
     + email
     + phone
     + account\_id (Foreign Key)
4. **Table: Guest**
   * Columns:
     + guest\_id (Primary Key)
     + person\_id (Foreign Key)
     + total\_rooms\_checked\_in
5. **Table: Receptionist**
   * Columns:
     + receptionist\_id (Primary Key)
     + person\_id (Foreign Key)
6. **Table: Server**
   * Columns:
     + server\_id (Primary Key)
     + person\_id (Foreign Key)
7. **Table: HotelLocation**
   * Columns:
     + location\_id (Primary Key)
     + name
     + address\_id (Foreign Key)
8. **Table: Hotel**
   * Columns:
     + hotel\_id (Primary Key)
     + name
9. **Table: Room**
   * Columns:
     + room\_id (Primary Key)
     + room\_number
     + style
     + status
     + booking\_price
     + is\_smoking
     + location\_id (Foreign Key)
10. **Table: RoomKey**
    * Columns:
      + key\_id (Primary Key)
      + barcode
      + is\_active
      + is\_master
      + room\_id (Foreign Key)
11. **Table: RoomBooking**
    * Columns:
      + booking\_id (Primary Key)
      + reservation\_number
      + start\_date
      + duration\_in\_days
      + status
      + check\_in
      + check\_out
      + guest\_id (Foreign Key)
      + room\_id (Foreign Key)
      + invoice\_id (Foreign Key)
12. **Table: Invoice**
    * Columns:
      + invoice\_id (Primary Key)
      + invoice\_number
      + generated\_at
13. **Table: InvoiceItem**
    * Columns:
      + item\_id (Primary Key)
      + description
      + price
      + quantity
      + invoice\_id (Foreign Key)

### 6.5.2 Database design structure