Hotel Management System Documentation

1. Introduction

This document outlines the design and implementation of a Hotel Management System built using Spring Boot. It covers the system architecture, modules, technologies used, and future enhancements.

2. Technologies Used

- Spring Boot: Framework for building RESTful backend services

- Java 21: Programming language used for development

- MySQL: Relational database for persistent storage

- JPA (Hibernate): ORM tool for database operations

- Swagger: API documentation and testing

- Maven: Build and dependency management

3. System Architecture

The application follows a layered architecture:

- Controller Layer: Manages HTTP requests and responses

- Service Layer: Contains business logic and validations

- Repository Layer: Interfaces with the database using JPA

- Model Layer: Defines entities and relationships

4. Module Descriptions

4.1 Hotel Module

Handles hotel creation, updates, deletion, and retrieval. Each hotel can have multiple amenities.

4.2 Room Module

Manages room details including type, location, and availability. Rooms are linked to hotels and amenities.

4.3 Room Type Module

Defines categories of rooms such as Deluxe, Suite, and Standard. Helps in filtering and pricing.

4.4 Amenity Module

Stores amenities like Wi-Fi, AC, and TV. Supports CRUD operations and links to hotels and rooms.

4.5 Hotel Amenity & Room Amenity Modules

Manages many-to-many relationships between hotels/rooms and amenities using join tables.

4.6 Reservation Module

Handles booking operations, including date-range filtering and reservation updates.

4.7 Payment Module

Records payments, calculates total revenue, and filters by payment status.

4.8 Review Module

Allows users to submit reviews and ratings for their stay. Reviews are linked to reservations.

5. API Design

The system exposes RESTful endpoints for each module. Standard HTTP methods (GET, POST, PUT, DELETE) are used. Path variables and query parameters enable dynamic filtering and access.

6. Database Schema

The schema is normalized with foreign key relationships. Join tables like HotelAmenity and RoomAmenity manage many-to-many mappings. Each entity has a primary key and relevant attributes.

7. Challenges Faced

- Managing complex entity relationships and join tables

- Implementing date-based filtering for reservations

- Ensuring data validation and consistency across layers

- Designing scalable and maintainable REST APIs

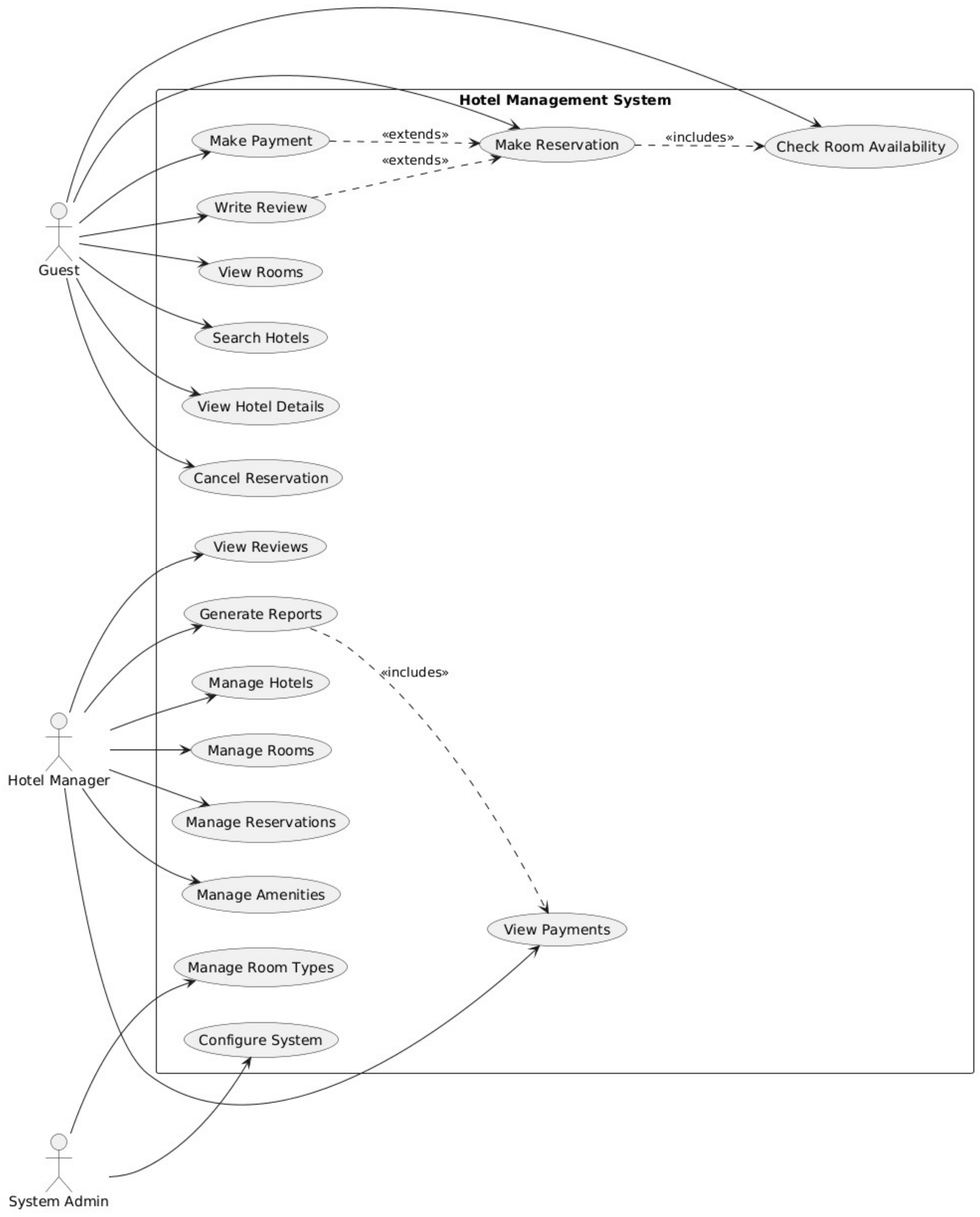
8. Future Scope

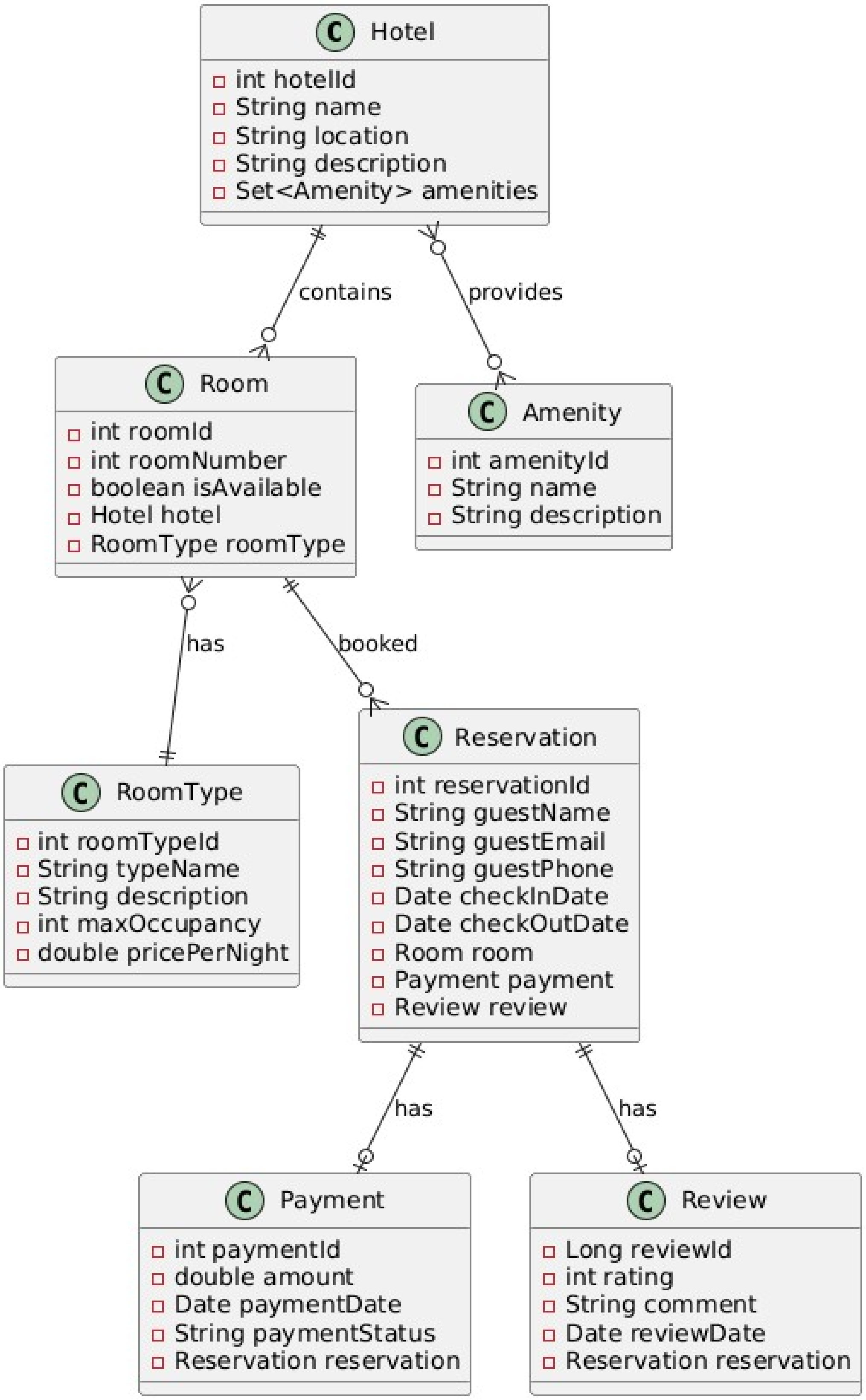
- Integration with third-party payment gateways

- Role-based authentication and authorization

- Mobile app integration for customer access

- Reporting dashboards for hotel analytics


Hotel Management System
Make Payment
«extends»
Make Reservation
«includes»
Check Room Availability
«extends»
Write Review
View Rooms
Search Hotels
View Hotel Details
Cancel Reservation
View Reviews
Generate Reports
«includes»
Manage Hotels
Manage Rooms
Manage Reservations
Manage Amenities
View Payments
Manage Room Types
Configure System
Guest
Hotel Manager
System Admin


Hotel
int hotelId
String name
String location
String description
Set<Amenity> amenities
contains
provides
Room
int roomId
int roomNumber
boolean isAvailable
Hotel hotel
RoomType roomType
Amenity
int amenityId
String name
String description
has
booked
RoomType
int roomTypeId
String typeName
String description
int maxOccupancy
double pricePerNight
Reservation
int reservationId
String guestName
String guestEmail
String guestPhone
Date checkInDate
Date checkOutDate
Room room
Payment payment
Review review
has
has
Payment
int paymentId
double amount
Date paymentDate
String paymentStatus
Reservation reservation
Review
Long reviewId
int rating
String comment
Date reviewDate
Reservation reservation