Software Design Document

Hotel Reservation System

Author - Ishika Gupta

1. System Overview

This section details the system architecture for the Hotel Management Platform, built on a microservices foundation to achieve high scalability, fault tolerance, and developer autonomy. The architecture leverages Spring Boot for service implementation, Eureka for service discovery, an API Gateway for centralized access control, and Apache Kafka as the backbone for event-driven workflows.

1.1. Architectural Overview

The system adopts a **Microservices Architecture**, where the application is decomposed into a collection of small, autonomous services, each running in its own process and communicating via lightweight mechanisms.

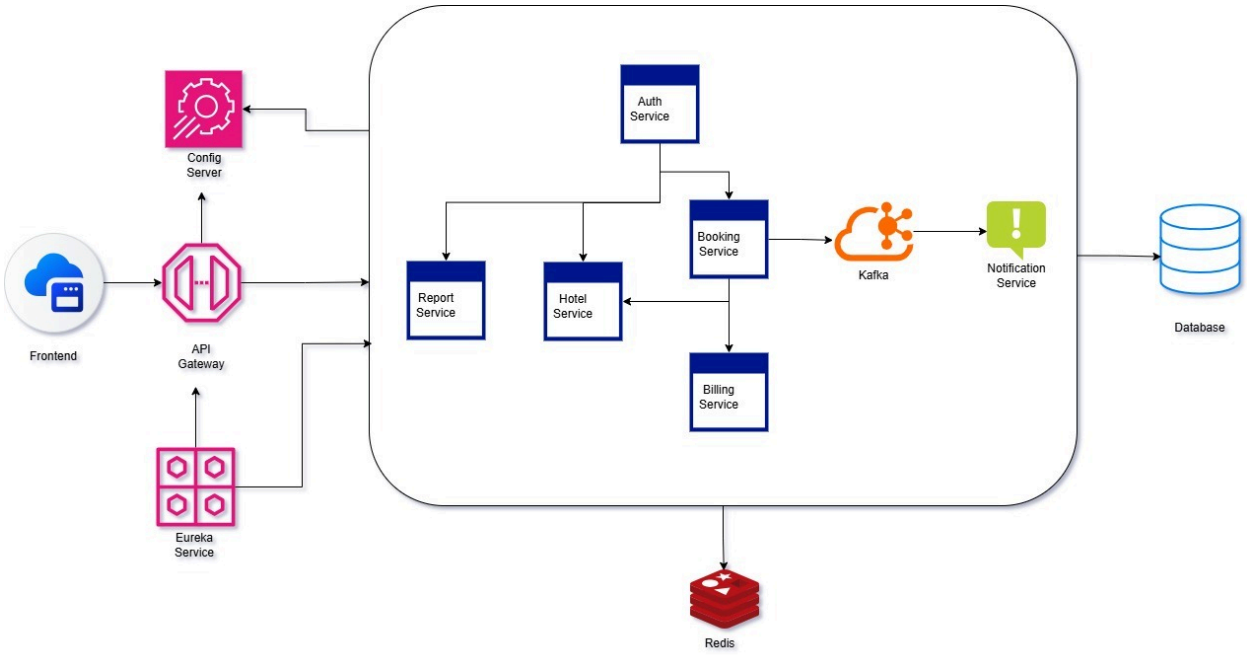

Fig 1.1. HMS Architecture Flow Diagram

2. Microservices Design

2.1. Auth Service

2.1.1 Purpose

To manage user identity, authentication, authorization, and role-based access control for the entire system.

2.1.2 Responsibilities

- Authenticate users and issue JWT tokens
- Support guest self-registration
- Manage system roles (ADMIN, MANAGER, RECEPTIONIST, GUEST)
- Create and manage staff accounts
- Assign staff to hotels
- Activate or deactivate user accounts

2.1.3 Databases

- Users
- Roles
- User–Hotel assignment mapping

2.1.4 Business Rules

- Only ADMIN can create and assign staff
- Guests can self-register
- Disabled users cannot authenticate
- Each staff member can be assigned to only one hotel

2.1.5 Key API Endpoints

Method	Endpoint	Role	Purpose
POST	/auth/register/guest	PUBLIC	Guest self-registration
POST	/auth/login	ALL	Login and get JWT
GET	/auth/me	ALL	Get logged-in user details
GET	/auth/admin/users	ADMIN	List all users
POST	/auth/admin/staff-allotment	ADMIN	Create staff and assign hotel
PUT	/auth/admin/staff/{userId}/hotel-allotment	ADMIN	Assign/reassign staff to hotel
PATCH	/auth/admin/users/{userId}/deactivate	ADMIN	Deactivate user
POST	/auth/change-password	ALL	Change Password

2.2 Hotel Service

2.2.1 Purpose

To serve as the single source of truth for hotel data, room configuration, inventory, pricing, and availability.

2.2.2 Responsibilities

- Manage hotels and hotel metadata (city, type)
- Manage room categories and pricing
- Maintain room inventory capacity
- Maintain per-day room availability
- Process atomic room holds and releases
- Track physical room operational state

2.2.3 Databases

- Hotels
- Room Categories
- Inventory
- Room Availability (per day)
- Physical Rooms

2.2.4 Business Rules

- Availability is category-based and date-based
- Inventory defines maximum capacity
- Physical room state does not control availability
- Holds must be atomic and time-bound
- Overbooking must be impossible

2.2.5. Key API Endpoints

A. Hotel Management

Method	Endpoint	Role	Purpose
POST	/hotels	ADMIN	Create hotel
PUT	/hotels/{hotelId}	ADMIN	Update hotel details
GET	/hotels	ALL	List hotels (filter by city)
GET	/hotels/{hotelId}	ALL	Get hotel details
DELETE	/hotels/{hotelId}	ADMIN	Disable hotel

B. Room Inventory Management

Method	Endpoint	Role	Purpose
PUT	/hotels/{hotelId}/inventory	MANAGER	Add/update inventory
GET	/hotels/{hotelId}/inventory	ADMIN, MANAGER	View inventory

C. Room Category Management

Method	Endpoint	Role	Purpose
POST	/hotels/{hotelId}/room-categories	ADMIN, MANAGER	Create room category
PUT	/room-categories/{categoryId}	ADMIN, MANAGER	Update room category
GET	/hotels/{hotelId}/room-categories	ALL	List room category
GET	/room-categories/{categoryId}	ALL	Room category details

D. Availability

Method	Endpoint	Role	Purpose
GET	/hotels/availability	ALL	Check availability by date & category

E. Public Hotel Search Api

Method	Endpoint	Role	Purpose
GET	public/hotels/search	None	Search Hotel by date and time

F. Physical Room Management

Method	Endpoint	Role	Purpose
POST	/hotels/{hotelId}/physical-rooms	ADMIN	Register rooms
PUT	/physical-rooms/{roomId}/assign	RECEPTIONIST	Assign room at check-in
PUT	/physical-rooms/{roomId}/release	RECEPTIONIST	Release room at checkout
PUT	/physical-rooms/{roomId}/maintenance	MANAGER	Mark maintenance
GET	/physical-rooms/{roomId}	ALL	View room state

G. Hold Apis

Method	Endpoint	Role	Purpose
POST	/hotels/holds	SYSTEM	Create room hold
POST	/hotels/holds/{holdId}/release	SYSTEM	Release hold

2.3 Booking Service

2.3.1 Purpose

To manage the complete booking lifecycle without owning inventory or availability data.

2.3.2 Responsibilities

- Create online and walk-in bookings
- Manage guest details per booking
- Handle booking confirmation, cancellation, and no-shows
- Orchestrate check-in and check-out workflows
- Expose internal booking data for billing and reporting

2.3.3 Databases

- Bookings
- Booking Guests
- Booking Status History

2.3.4 Business Rules

- Booking service must not modify availability directly
- Each booking must reference a valid hold
- Guest details are mandatory before check-in
- Walk-in bookings must respect availability
- Booking state transitions are strictly enforced

2.3.5 Key API Endpoints

A. Booking Management

Method	Endpoint	Role	Purpose
POST	/reservations	GUEST, RECEPTIONIST	Create booking (after hold)
GET	/reservations/{id}	ALL	Get booking details
GET	/reservations/my-booking	GUEST	Get user bookings
GET	/reservations/hotel/{hotelId}	ADMIN, MANAGER	Get hotel bookings
DELETE	/reservations/{id}	GUEST	Cancel booking
PUT	/reservations/{id}	GUEST	Update booking
GET	/reservations/{id}/guests	ALL	Get Guests Details for booking
PUT	/reservations/{id}/guests	GUEST	Update Guest Detail for booking
POST	/reservations/{id}/guests	RECEPTIONIST	Add walk-in guests

B. LifeCycle Management

Method	Endpoint	Role	Purpose
POST	/reservations/{id}/confirm	GUEST	Confirm booking
POST	/reservations/{id}/check-in	RECEPTIONIST	Check-in guest
POST	/reservations/{id}/check-out	RECEPTIONIST	Check-out guest

C. Internal Management

Method	Endpoint	Role	Purpose
POST	/internal/reservations/booked-room	INTERNAL(Hotel Service)	Get Rooms Booked By date
GET	/internal/reservations/start-tomorrow	INTERNAL(Notification Service)	Reminder
GET	/internal/reservations/{id}/summary	INTERNAL(Billing Service)	Summary for Booking

2.3 Billing Service

2.3.1. Purpose

To manage all financial operations related to bookings.

2.3.2. Responsibilities

- Generate invoices for bookings
- Track payment status
- Support online and offline payment modes
- Maintain billing records

2.3.3 Databases

- Bills
- Payments

2.3.4 Business Rules

- Billing data must be derived from Booking Service
- A booking can have only one active bill

2.3.5 Key API Endpoints

A. Bill Generation

Method	Endpoint	Role	Purpose
POST	/bills/generate/{bookingId}	SYSTEM	Generate bill
GET	/bills/{bookingId}	ALL	Get bill details

B. Payment

Method	Endpoint	Role	Purpose
POST	/payments	SYSTEM	Record Payment for invoice
GET	/payments/invoice/{invoiceId}	ALL	Get bill details

2.4 Notification Service

2.4.1 Purpose

To send asynchronous notifications to users and staff based on system events.

2.4.2 Responsibilities

- Consume events from Kafka
- Send booking, payment, and reminder notifications
- Support email and future notification channels

2.4.3 Databases

- Notification logs (optional)

2.4.4 Business Rules

- Notifications are event-driven only
- Failure to send a notification must not affect core flows
- Notifications must be idempotent

2.4.5 Key API Endpoints

Method	Endpoint	Role	Purpose
POST	/notifications/send	GUEST	Create booking (after hold)
GET	/notifications/user/{userId)	ALL	Get booking details
GET	/notifications/{id}	GUEST	Get user bookings

2.5 Report Service

2.5.1 Purpose

To provide aggregated, read-only analytical insights and operational reports across hotels, bookings, and revenue for administrative and managerial decision-making.

2.5.2 Responsibilities

- Generate booking summary reports
- Generate revenue and payment reports
- Provide occupancy and utilization metrics
- Support hotel-level and system-wide analytics
- Expose read-only reporting APIs for dashboards

2.5.5 Databases

- Reporting database (read-optimized)
- Materialized views or denormalized tables populated from events

2.5.4 Business Rules

- Reporting data is read-only
- Reports must not affect transactional workflows
- Data is eventually consistent
- Reports are generated from events or replicated data

- Access is restricted based on role (ADMIN, MANAGER)

2.5.5 Key API Endpoints

Method	Endpoint	Role	Purpose
GET	/reports/bookings	ADMIN	Booking summary report
GET	/reports/revenue	ADMIN	Revenue report
GET	/reports/occupancy	ADMIN, MANAGER	Occupancy report
GET	/reports/hotel/{hotelId}	ADMIN	Hotel-level analytics