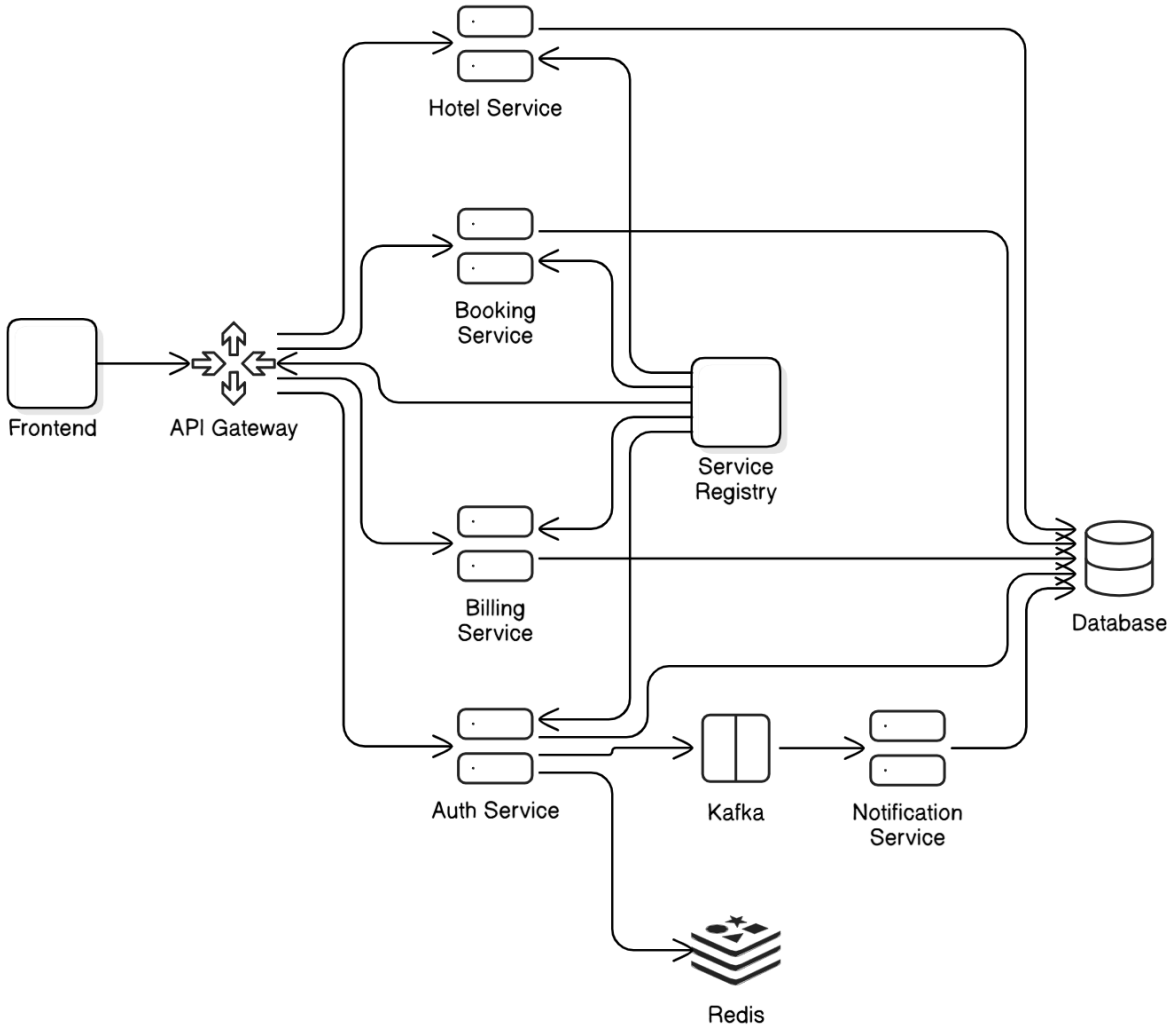
Design



eraser

Security-Workflow

1.User opens the frontend and enters username and password.
2.Frontend sends login request to the API Gateway.
3.API Gateway forwards the login request to the auth-service because it is a public route.
4.Auth-service verifies credentials from the database.
5.If credentials are valid, auth-service generates a JWT containing userId, role, and hotelId.
6.Auth-service sends the JWT back to the frontend through the gateway.
7.Frontend stores the JWT (for example in memory or local storage).

8.For any protected action, frontend sends a request to the API Gateway with `Authorization: Bearer <JWT>`.
9.API Gateway checks if the route is public or protected.
10.If protected, the gateway validates the JWT signature and expiry.
11.If the token is invalid or missing, the gateway immediately returns 401 to the client.
12.If the token is valid, the gateway extracts userId, role, and hotelId from the JWT.
13.The gateway removes the JWT and adds these values as internal headers.
14.The gateway routes the request to the correct service.
15.The service reads the headers sent by the gateway (userId, role, hotelId).
16.The service checks **if the role is allowed** to perform that action.
17.If the role is not allowed, the service returns **403 Forbidden**.
18.If the role is allowed, the service then checks **business rules** (ownership, hotelId match, booking ownership, etc.).
19.If any business rule fails, the service returns **403 Forbidden**.
20.If everything passes, the service processes the request and returns success

---

1.CASE :Search->Book flow

Frontend sends a **request** to API Gateway

```
Get /api/hotels/search?city=Mumbai → public endpoint
Hotel-Service: - Queries hotels table - Returns list of hotels with basic info Response: [Hotel 1, Hotel 2, Hotel 3...]
// response payload missing-TODO
```


```
{
    "success": true,
    "message": "Hotels search completed successfully",
    "data": [
        {
            "id": 3,
            "name": "Grand Plaza Hotel",
            "description": "Luxury hotel in Mumbai",
            "address": "123 Marine Drive",
            "city": "Mumbai",
            "state": "Maharashtra",
            "country": "India",
            "pincode": "400001",
            "contactNumber": "9876543210",
            "email": "contact@grandplaza.com",
            "starRating": 5,
            "amenities": "WiFi,Pool,Gym,Spa,Restaurant",
            "imageUrl": null,
            "status": "ACTIVE",
            "totalRooms": 2,
            "availableRooms": 2,
```

```
            "createdAt": "2025-12-30T22:04:05.685103",
            "updatedAt": "2025-12-30T22:10:25.264644"
        }
    ]
}
```


Step 2: View Hotel Details
User: Clicks "Grand Plaza Hotel" (id=1) Frontend → API Gateway → Hotel-Service: GET /api/hotels/1 Hotel-Service: - Returns hotel details

Response: {name, address, rating, totalRooms, amenities...}

Step 3: User Enters Dates
Frontend → API Gateway → Booking-Service: GET /api/bookings/check-availability?hotelId=1&checkIn=2024-12-02&checkOut=2024-12-03
Booking-Service:
Step 3a: Calls Hotel-Service (OpenFeign) GET /api/rooms/hotel/1 Gets: [Room 101, Room 102, Room 103]

Step 3b: Queries own DB SELECT room_id FROM bookings WHERE hotel_id=1 AND dates overlap Gets: [Room 101] (booked)

Step 3c: Filters Available = All rooms - Booked rooms - (status != AVAILABLE) Result: [Room 102, Room 103]

Step 5: Create Booking
User: Clicks "Confirm Booking"

Frontend → API Gateway → Booking-Service:
POST /api/bookings


```
Body: {
  userId: 5,
  hotelId: 1,
  roomId: 102,
  checkIn: "2024-12-02",
  checkOut: "2024-12-03",
  totalAmount: 5000
}
```

```
Booking-Service:
  Step 5a: Double-check availability (prevent race condition)

  Step 5b: Create booking in bookings table
    status = PENDING

  Step 5c: Publish Kafka event
    Event: booking-created
    Payload: {bookingId, userId, roomId, dates...}

Response: {bookingId: 123, status: PENDING, message: "Booking confirmed"}
```

Step 6: Background Processing (Kafka)

```
Event: booking-created

Notification-Service (listens):
  - Sends email to user: "Booking confirmed"
  - Sends SMS to user

Payment-Service (listens):
  - Process payment
  - If success → publish payment-completed event

(Hotel-Service does NOT listen yet - waits for check-in event)
```

---

2. User makes a booking

User selects a hotel and room details on frontend
hotelId, roomType, check-in date, check-out date, number of guests.
Frontend sends a **POST request** to API Gateway

```
POST /api/bookings
```

```
{
  "hotelId": 5,
  "roomType": "DELUXE",

  "checkInDate": "2025-01-10",
  "checkOutDate": "2025-01-12",
  "guests": 2,
  "rooms": 1
}
```

```
API Gateway receives the request.
This is a protected endpoint
Gateway validates JWT
Extracts userId, role, hotelId
Adds headers:

 X-User-Id
X-User-Role
X-Hotel-Id
Routes request to Booking Service
```

Ownership

- Hotel Service owns, hotel existence, hotel status (ACTIVE / INACTIVE), room types / capacity (static data)
- Booking Service owns ,bookings , availability calculations, date overlaps, room inventory usage

Booking Service receives the request.

Reads `X-User-Id` and `X-User-Role`

Checks role:

GUEST → allowed , MANAGER / ADMIN → allowed (depends on rules)
If role not allowed → return 403
Booking Service performs validates hotel + room definition

```
GET /api/internal/hotels/{hotelId}
```

```
{
  "hotelId": 5,
  "status": "ACTIVE",
 "roomTypes": ["DELUXE":10, "STANDARD":20]
}
```

Booking Service checks availability , queries own db to check available

If available < requested rooms

```
{  "message": "Rooms not available for selected dates"}
```

8. Booking Service creates booking record
   Database entry:
   bookingId
9. Userid
10. hotelId
11. roomType
12. Dates
13. status = CONFIRMED totalAmount

Booking publishes a booking-created event

```
{
  "bookingId": 789,
  "userId": 123,
  "hotelId": 5,
  "amount": 4200
}
```

Response to client

```
{
 "bookingId": 789,
 "status": "CONFIRMED",
 "totalAmount": 4200
}
```

CHECK-IN FLOW (inside Booking Service)

9. On check-in day, (receptionist) calls:

`POST /api/bookings/{bookingId}/check-in in gateway`
`After validations and checks`

booking Service publishes: `booking-checked-in (event)`

booking Service publishes: `booking-completed (event)`

1. Auth Service – endpoints + roles

Auth is **open + self-contained**. Other services trust JWT.

Endpoints

Method	Endpoint	Access	Purpose
POST	`/auth/register`	PUBLIC	Guest signup
POST	`/auth/login`	PUBLIC	Login, get JWT
PUT	`/auth/me/username`	AUTHENTICATED	Update username

GET	`/auth/users`	ADMIN	List users
POST	`/auth/create-user`	ADMIN	Create manager and receptionist

Roles:

- PUBLIC → no token
- AUTHENTICATED → any logged-in user
- ADMIN → system admin only

---

2. Hotel Service – endpoints + roles

Hotel service owns **hotels + rooms + pricing + physical room state**

Hotel APIs

Method	Endpoint	Description	Access Control
POST	/api/hotels	Create a new hotel	ADMIN only
GET	/api/hotels/{id}	Get hotel details by ID	All authenticated users
GET	/api/hotels/search	Search hotels by city, state, rating, etc.	All authenticated users

PUT	/api/hotels/{id}	Update hotel details	ADMIN, MANAGER (own hotel only)
POST	/api/rooms	Create a new room in a hotel	ADMIN, MANAGER (own hotel only)
GET	/api/rooms/{id}	Get room details by ID	All authenticated users
GET	/api/rooms/hotel/{hotelId}	Get all rooms for a specific hotel	All authenticated users
PUT	/api/rooms/{id}	Update room details	ADMIN, MANAGER (own hotel only), RECEPTIONIST (own hotel only)
PATCH	/api/rooms/{id}/status	Update specifically the room status (e.g., to CLEANING)	ADMIN, MANAGER (own hotel only), RECEPTIONIST (own hotel only)
DELETE	/api/rooms/{id}	Delete a room	ADMIN, MANAGER (own hotel only)

- Hotel service never stores booking data

Request and response

POST /api/hotels
Role: ADMIN

```
{
  "name": "Grand Plaza Hotel",
  "description": "Luxury hotel in Mumbai",
  "address": "123 Marine Drive",
  "city": "Mumbai",
  "state": "Maharashtra",
  "country": "India",          // will implement google maps pin later
  "pincode": "400001",
  "contactNumber": "9876543210",
  "email": "contact@grandplaza.com",
  "starRating": 5,
  "amenities": "WiFi,Pool,Gym,Spa,Restaurant",
  "status": "ACTIVE"
}

Response -success
{
Hotel-id:3
}
```

GET /api/hotels/ –admin get all hotels

```
[
{hotel1},{hotel2},...  ]
```

POST /{{baseurl}}/api/hotels/rooms
Role: ADMIN / MANAGER

```
 {
  "hotelId": 3,
  "roomNumber": "101",
  "roomType": "DELUXE",
  "pricePerNight": 5000.00,
  "maxOccupancy": 2,
  "floorNumber": 1,
  "bedType": "King",
  "roomSize": 400,
  "amenities": "AC,TV,WiFi,Mini Bar",
  "description": "Spacious deluxe room with city view",
```

```
  "status": "AVAILABLE",
  "isActive": true
}
```

```
ON INVALID ROOM NO
{
    "success": false,
    "message": "Room number 101 already exists for this hotel"
   }
```

---

3. Booking Service – endpoints + roles

Booking service owns **availability logic + lifecycle**

Booking APIs

Method	Endpoint	Description	Access Control
GET	/api/bookings/availability	Check room availability	All authenticated users

POST	/api/bookings	Create a new booking	GUEST, ADMIN
GET	/api/bookings/{id}	Get booking details by ID	Owner (GUEST), Hotel staff, ADMIN
GET	/api/bookings/my-bookings	Get all bookings for the current guest	GUEST only
GET	/api/bookings/hotel/{hotelId}	Get all bookings for a specific hotel	MANAGER, RECEPTIONIST (own hotel only), ADMIN
GET	/api/bookings	Get all bookings (system-wide)	ADMIN only
PATCH	/api/bookings/{id}/cancel	Cancel an existing booking	Owner (GUEST), Hotel staff, ADMIN
POST	/api/bookings/{id}/check-in	Check-in a guest	MANAGER, RECEPTIONIST, ADMIN
POST	/api/bookings/{id}/check-out	Check-out a guest	MANAGER, RECEPTIONIST, ADMIN

GET	/api/bookings/hotel/{hotelId}/today-checkins	Get today's check-ins for a hotel	MANAGER, RECEPTIONIST, ADMIN
GET	/api/bookings/hotel/{hotelId}/today-checkouts	Get today's check-outs for a hotel	MANAGER, RECEPTIONIST, ADMIN

Booking lifecycle rules (enforced here)

State	Who can trigger
BOOKED	Guest
CANCELLED	Guest / system
CHECKED_IN	Receptionist
CHECKED_OUT	Receptionist
GUEST_UNAVAIlable	Scheduled job

---

4. Billing Service – endpoints + roles

Billing is **derived data only**

APIs

Method	Endpoint	Roles	Purpose
POST	`/bills/generate/{bookingId}`	INTERNAL	Generate bill
GET	`/bills/{bookingId}`	GUEST, MANAGER, ADMIN	View bill

PUT	`/bills/{bookingId}/mark-paid`	ADMIN	Mark as paid

Triggered automatically on:

- CHECKED_OUT

No guest payment logic.

---

5. Notification Service – endpoints + roles

APIs

Method	Endpoint	Access	Purpose
POST	`/notifications/booking-confirmation`	INTERNAL	Booking email
POST	`/notifications/reminder`	INTERNAL	Reminder email

Plus:

- `@Scheduled` job for no-show reminders & check-in reminders

Endpoints and payload

Register user:

POST http://localhost:8081/api/auth/register
Content-Type: application/json

```
{
"username": "john_guest",
"email": "john@example.com",
"password": "password123",
```

```
 "fullName": "John Doe",
 "phoneNumber": "9876543210"
  }
```

role is ignored, always creates GUEST

Admin login – (admin is pre seeded in the db)

POST http://localhost:8081/api/auth/login
Content-Type: application/json

```
{ "username": "admin", "password": "admin123" }
```

Response:

```
{
  "token": "eyJhbGciOiJIUzI1NiJ9...",
  "tokenType": "Bearer",
  "user": {
    "id": 1,
    "username": "admin",
    "role": "ADMIN",
    ...
  }
}
```

```

---

**. admin creates manager (protected):****

```

POST http://localhost:8081/api/auth/create-user
Content-Type: application/json
Authorization: Bearer <admin-token>

```
{
 "username": "manager_hotel1",
  "email": "manager1@hotel.com",
```

```
 "password": "manager123", "role":
 "MANAGER", "hotelId": 1,
"fullName": "Hotel Manager One",
"phoneNumber": "1234567890"
 }


{ "username": "receptionist_hotel1",
 "email": "receptionist1@hotel.com",
 "password": "receptionist123",
 "role": "RECEPTIONIST", "hotelId": 1,
 "fullName": "Receptionist One",
"phoneNumber": "5555555555" }
```

---

validation checks:** - guest registration → no hotelId needed - manager/receptionist creation → hotelId required - only admin can call /create-user endpoint - if non-admin tries → 403 forbidden

---

workflow:

1. admin logs in → gets JWT token
2. admin creates hotels (in hotel-service) → hotels get IDs
3. admin creates managers/receptionists → selects hotel from dropdown (client shows hotel name + ID)
4. manager logs in → can only access their assigned hotel

JWT

Identity → `userId` (immutable, DB-backed)

Authorization → `role`

Multi-tenancy → `hotelId (JWT carries identity, role, and hotel context so downstream services stay stateless.)`

Stateless → no DB lookup needed per request – stateless

Hotel Management System

---

1. Authentication

All API endpoints require authentication through the API Gateway. The gateway validates JWT tokens and forwards requests with the following headers:

Required Headers:

```
X-User-Id: {userId}
X-Username: {username}
X-User-Email: {email}
X-User-Role: {ADMIN|MANAGER|RECEPTIONIST|GUEST}
X-Hotel-Id: {hotelId}
```

---

2. Hotel Service API Endpoints

3.1 Create Hotel

Endpoint: POST /api/hotels

Access Control: ADMIN only

Request Body:

```
{
  "name": "Grand Plaza Hotel",
  "description": "Luxury hotel in downtown area",
  "address": "123 Main Street",
  "city": "New York",
  "state": "NY",
  "country": "USA",
  "zipCode": "10001",
  "phone": "+1234567890",
  "email": "contact@grandplaza.com",
  "totalRooms": 150,
  "amenities": ["WiFi", "Pool", "Gym", "Restaurant"]
}
```

```
Response Body:
{
  "success": true,
  "message": "Hotel created successfully",
  "data": {
    "id": 1,
    "name": "Grand Plaza Hotel",
    "description": "Luxury hotel in downtown area",
    "address": "123 Main Street",
    "city": "New York",
    "state": "NY",
    "country": "USA",
    "zipCode": "10001",
    "phone": "+1234567890",
    "email": "contact@grandplaza.com",
    "totalRooms": 150,
    "availableRooms": 150,
    "status": "ACTIVE",
    "rating": 0.0,
    "amenities": ["WiFi", "Pool", "Gym", "Restaurant"],
    "createdAt": "2025-01-15T10:30:00",
    "updatedAt": "2025-01-15T10:30:00"
  }
}
```

Status Codes:

- 201: Hotel created successfully
- 400: Validation error
- 403: Unauthorized access
- 500: Internal server error

3.2 Get Hotel by ID

Endpoint: GET /api/hotels/{id}

Access Control: All authenticated users

Path Parameters:

- id: Hotel identifier (Long)

Response Body:


```
{
  "success": true,
  "message": "Hotel retrieved successfully",
  "data": {
    "id": 1,
    "name": "Grand Plaza Hotel",
    "description": "Luxury hotel in downtown area",
    "address": "123 Main Street",
    "city": "New York",
    "state": "NY",
    "country": "USA",
    "zipCode": "10001",
    "phone": "+1234567890",
    "email": "contact@grandplaza.com",
    "totalRooms": 150,
    "availableRooms": 145,
    "status": "ACTIVE",
    "rating": 4.5,
    "amenities": ["WiFi", "Pool", "Gym", "Restaurant"],
    "createdAt": "2025-01-15T10:30:00",
    "updatedAt": "2025-01-15T10:30:00"
  },
}
```


Status Codes:

- 200: Success
- 404: Hotel not found
- 500: Internal server error

3.3 Search Hotels

Endpoint: GET /api/hotels/search

Access Control: All authenticated users

Query Parameters:

- city: City name (String, optional)
- state: State code (String, optional)
- minRating: Minimum rating (Double, optional)

- status: Hotel status (String, optional)

Example: GET /api/hotels/search?city=New York

Response Body:


```
{
  "success": true,
  "message": "Hotels retrieved successfully",
  "data": [
    {
      "id": 1,
      "name": "Grand Plaza Hotel",
      "city": "New York",
      "state": "NY",
      "country": "USA",
      "totalRooms": 150,
      "availableRooms": 145,
      "status": "ACTIVE",
      "rating": 4.5,
      "amenities": ["WiFi", "Pool", "Gym", "Restaurant"]
    },
    {
      "id": 2,
      "name": "Royal Suites",
      "city": "New York",
      "state": "NY",
      "country": "USA",
      "totalRooms": 200,
      "availableRooms": 180,
      "status": "ACTIVE",
      "rating": 4.8,
      "amenities": ["WiFi", "Spa", "Restaurant"]
    }
  ],
  "timestamp": "2025-01-15T11:00:00"
}
```


Status Codes:

- 200: Success
- 400: Invalid parameters
- 500: Internal server error

3.4 Update Hotel

Endpoint: PUT /api/hotels/{id}

Access Control: ADMIN, MANAGER (own hotel only)

Request Body:

```
{
  "name": "Grand Plaza Hotel & Spa",
  "description": "Luxury hotel with spa facilities",
  "phone": "+1234567891",
  "email": "info@grandplaza.com",
  "amenities": ["WiFi", "Pool", "Gym", "Restaurant", "Spa"]
}
```

Response Body:

```
{
  "success": true,
  "message": "Hotel updated successfully",
  "data": {
    "id": 1,
    "name": "Grand Plaza Hotel & Spa",
    "description": "Luxury hotel with spa facilities",
    "address": "123 Main Street",
    "city": "New York",
    "state": "NY",
    "country": "USA",
    "zipCode": "10001",
    "phone": "+1234567891",
    "email": "info@grandplaza.com",
    "totalRooms": 150,
    "availableRooms": 145,
    "status": "ACTIVE",
    "rating": 4.5,
    "amenities": ["WiFi", "Pool", "Gym", "Restaurant", "Spa"],
    "createdAt": "2025-01-15T10:30:00",
    "updatedAt": "2025-01-15T12:00:00"
  },
```

```
}
```

Status Codes:

- 200: Hotel updated successfully
- 400: Validation error
- 403: Unauthorized access
- 404: Hotel not found
- 500: Internal server error

3.5 Create Room

Endpoint: POST /api/rooms

Access Control: ADMIN, MANAGER (own hotel only)

Request Body:

```
{
  "hotelId": 1,
  "roomNumber": "101",
  "roomType": "DELUXE",
  "pricePerNight": 150.00,
  "description": "Spacious deluxe room with city view",
  "maxOccupancy": 2,
  "amenities": ["TV", "Mini Bar", "Air Conditioning"]
}
```

Response Body:

```
{
  "success": true,
  "message": "Room created successfully",
  "data": {
    "id": 1,
    "hotelId": 1,
    "roomNumber": "101",
    "roomType": "DELUXE",
    "pricePerNight": 150.00,
```

```
    "status": "AVAILABLE",
    "isActive": true,
    "description": "Spacious deluxe room with city view",
    "maxOccupancy": 2,
    "amenities": ["TV", "Mini Bar", "Air Conditioning"],
    "createdAt": "2025-01-15T13:00:00",
    "updatedAt": "2025-01-15T13:00:00"
  },
  "timestamp": "2025-01-15T13:00:00"
}
```


Status Codes:

- 201: Room created successfully
- 400: Validation error
- 403: Unauthorized access
- 404: Hotel not found
- 500: Internal server error

3.7 Get Room by ID

Endpoint: GET /api/rooms/{id}

Access Control: All authenticated users

Path Parameters:

- id: Room identifier (Long)

Response Body:


```
{
  "success": true,
  "message": "Room retrieved successfully",
  "data": {
    "id": 1,
    "hotelId": 1,
    "roomNumber": "101",
    "roomType": "DELUXE",
    "pricePerNight": 150.00,
    "status": "AVAILABLE",
    "isActive": true,
```

```
    "description": "Spacious deluxe room with city view",
    "maxOccupancy": 2,
    "amenities": ["TV", "Mini Bar", "Air Conditioning"],
    "createdAt": "2025-01-15T13:00:00",
    "updatedAt": "2025-01-15T13:00:00"
  },
  "timestamp": "2025-01-15T13:30:00"
}
```

Status Codes:

- 200: Success
- 404: Room not found
- 500: Internal server error

3.8 Get Rooms by Hotel ID

Endpoint: GET /api/rooms/hotel/{hotelId}

Access Control: All authenticated users

Path Parameters:

- hotelId: Hotel identifier (Long)

```
Response Body:
{
  "success": true,
  "message": "Rooms retrieved successfully",
  "data": [
    {
      "id": 1,
      "hotelId": 1,
      "roomNumber": "101",
      "roomType": "DELUXE",
      "pricePerNight": 150.00,
      "status": "AVAILABLE",
      "isActive": true,
      "maxOccupancy": 2
    },
    {
      "id": 2,
```

```
      "hotelId": 1,
      "roomNumber": "102",
      "roomType": "SUITE",
      "pricePerNight": 250.00,
      "status": "OCCUPIED",
      "isActive": true,
      "maxOccupancy": 4
    }
  ],

}
```


Status Codes:

- 200: Success
- 404: Hotel not found
- 500: Internal server error

3.9 Update Room

Endpoint: PUT /api/rooms/{id}

Access Control: ADMIN, MANAGER (own hotel only), RECEPTIONIST (own hotel only)

Path Parameters:

- id: Room identifier (Long)

Request Body:


```
{
  "roomNumber": "101A",
  "pricePerNight": 175.00,
  "description": "Updated deluxe room with premium amenities",
  "amenities": ["TV", "Mini Bar", "Air Conditioning", "Coffee Machine"]
}
```


Response Body:


```
{
  "success": true,
  "message": "Room updated successfully",
  "data": {
```

```
    "id": 1,
    "hotelId": 1,
    "roomNumber": "101A",
    "roomType": "DELUXE",
    "pricePerNight": 175.00,
    "status": "AVAILABLE",
    "isActive": true,
    "description": "Updated deluxe room with premium amenities",
    "maxOccupancy": 2,
    "amenities": ["TV", "Mini Bar", "Air Conditioning", "Coffee Machine"],
    "createdAt": "2025-01-15T13:00:00",
    "updatedAt": "2025-01-15T14:30:00"
  },
  "timestamp": "2025-01-15T14:30:00"
}
```


Status Codes:

- 200: Room updated successfully
- 400: Validation error
- 403: Unauthorized access
- 404: Room not found
- 500: Internal server error

3.10 Update Room Status

Endpoint: PATCH /api/rooms/{id}/status

Access Control: ADMIN, MANAGER (own hotel only), RECEPTIONIST (own hotel only)

Path Parameters:

- id: Room identifier (Long)

Query Parameters:

- status: New room status (AVAILABLE, OCCUPIED, CLEANING, MAINTENANCE)

Example: PATCH /api/rooms/1/status?status=OCCUPIED

Response Body:


```
{
  "success": true,
```

```
  "message": "Room status updated successfully",
  "data": {
    "id": 1,
    "hotelId": 1,
    "roomNumber": "101A",
    "roomType": "DELUXE",
    "pricePerNight": 175.00,
    "status": "OCCUPIED",
    "isActive": true,
    "description": "Updated deluxe room with premium amenities",
    "maxOccupancy": 2,
    "amenities": ["TV", "Mini Bar", "Air Conditioning", "Coffee Machine"],
    "createdAt": "2025-01-15T13:00:00",
    "updatedAt": "2025-01-15T15:00:00"
  },
  "timestamp": "2025-01-15T15:00:00"
}
```


Status Codes:

- 200: Status updated successfully
- 400: Invalid status
- 403: Unauthorized access
- 404: Room not found
- 500: Internal server error

3.11 Delete Room

Endpoint: DELETE /api/rooms/{id}

Access Control: ADMIN, MANAGER (own hotel only)

Path Parameters:

- id: Room identifier (Long)

Response Body:


```
{
  "success": true,
  "message": "Room deleted successfully",
  "data": null,
```

```
  "timestamp": "2025-01-15T15:30:00"
}
```

Status Codes:

- 200: Room deleted successfully
- 403: Unauthorized access
- 404: Room not found
- 500: Internal server error

---

4. Booking Service API Endpoints

4.1 Check Room Availability

Endpoint: GET /api/bookings/availability

Access Control: All authenticated users

Query Parameters:

- hotelId: Hotel identifier (Long, required)
- checkInDate: Check-in date (Date, required, format: YYYY-MM-DD)
- checkOutDate: Check-out date (Date, required, format: YYYY-MM-DD)

Example: GET /api/bookings/availability?hotelId=1&checkInDate=2025-01-20&checkOutDate=2025-01-25

Response Body:

```
{
  "success": true,
  "message": "Availability checked successfully",
  "data": {
    "hotelId": 1,
    "checkInDate": "2025-01-20",
    "checkOutDate": "2025-01-25",
    "totalRooms": 150,
    "availableRooms": 142,
    "availableRoomList": [
      {
        "id": 1,
        "hotelId": 1,
```

```
        "roomNumber": "101",
        "roomType": "DELUXE",
        "pricePerNight": 150.00,
        "status": "AVAILABLE",
        "isActive": true,
        "maxOccupancy": 2
      },
      {
        "id": 3,
        "hotelId": 1,
        "roomNumber": "103",
        "roomType": "SUITE",
        "pricePerNight": 250.00,
        "status": "AVAILABLE",
        "isActive": true,
        "maxOccupancy": 4
      }
    ]
  },
}
```


Status Codes:

- 200: Success
- 400: Invalid date range
- 404: Hotel not found
- 500: Internal server error

4.2 Create Booking

Endpoint: POST /api/bookings

Access Control: GUEST, ADMIN

Request Body:


```
{
  "hotelId": 1,
  "roomId": 1,
  "checkInDate": "2025-01-20",
  "checkOutDate": "2025-01-25",
  "guestName": "John Doe",
```

```
  "guestEmail": "john.doe@example.com",
  "guestPhone": "+1234567890",
  "numberOfGuests": 2,
  "specialRequests": "Late check-in requested"
}
```

Response Body:

```
{
  "success": true,
  "message": "Booking created successfully",
  "data": {
    "id": 1,
    "userId": 10,
    "hotelId": 1,
    "roomId": 1,
    "roomNumber": "101",
    "roomType": "DELUXE",
    "checkInDate": "2025-01-20",
    "checkOutDate": "2025-01-25",
    "totalAmount": 750.00,
    "status": "CONFIRMED",
    "guestName": "John Doe",
    "guestEmail": "john.doe@example.com",
    "guestPhone": "+1234567890",
    "numberOfGuests": 2,
    "numberOfNights": 5,
    "specialRequests": "Late check-in requested",
    "cancellationReason": null,
    "cancelledAt": null,
    "checkedInAt": null,
    "checkedOutAt": null,
    "createdAt": "2025-01-15T16:30:00",
    "updatedAt": "2025-01-15T16:30:00"
  },
  "timestamp": "2025-01-15T16:30:00"
}
```

Status Codes:

- 201: Booking created successfully
- 400: Validation error or room not available

- 403: Unauthorized access
- 404: Hotel or room not found
- 500: Internal server error

4.3 Get Booking by ID

Endpoint: GET /api/bookings/{id}

Access Control: Owner (GUEST), Hotel staff, ADMIN

Path Parameters:

- id: Booking identifier (Long)

Response Body:


```
{
  "success": true,
  "message": "Booking retrieved successfully",
  "data": {
    "id": 1,
    "userId": 10,
    "hotelId": 1,
    "roomId": 1,
    "roomNumber": "101",
    "roomType": "DELUXE",
    "checkInDate": "2025-01-20",
    "checkOutDate": "2025-01-25",
    "totalAmount": 750.00,
    "status": "CONFIRMED",
    "guestName": "John Doe",
    "guestEmail": "john.doe@example.com",
    "guestPhone": "+1234567890",
    "numberOfGuests": 2,
    "numberOfNights": 5,
    "specialRequests": "Late check-in requested",
    "cancellationReason": null,
    "cancelledAt": null,
    "checkedInAt": null,
    "checkedOutAt": null,
    "createdAt": "2025-01-15T16:30:00",
    "updatedAt": "2025-01-15T16:30:00"
  },
```

}

Status Codes:

- 200: Success
- 403: Unauthorized access
- 404: Booking not found
- 500: Internal server error

4.4 Get My Bookings

Endpoint: GET /api/bookings/my-bookings

Access Control: GUEST only

```
Response Body:
{
  "success": true,
  "message": "User bookings retrieved successfully",
  "data": [
    {
      "id": 1,
      "userId": 10,
      "hotelId": 1,
      "roomId": 1,
      "roomNumber": "101",
      "roomType": "DELUXE",
      "checkInDate": "2025-01-20",
      "checkOutDate": "2025-01-25",
      "totalAmount": 750.00,
      "status": "CONFIRMED",
      "guestName": "John Doe",
      "numberOfNights": 5,
      "createdAt": "2025-01-15T16:30:00"
    },
    {
      "id": 2,
      "userId": 10,
      "hotelId": 2,
      "roomId": 15,
      "roomNumber": "205",
      "roomType": "SUITE",
```

```
      "checkInDate": "2025-02-10",
      "checkOutDate": "2025-02-15",
      "totalAmount": 1250.00,
      "status": "PENDING",
      "guestName": "John Doe",
      "numberOfNights": 5,
      "createdAt": "2025-01-15T17:00:00"
    }
  ]
}
```

Status Codes:

- 200: Success
- 403: Unauthorized access
- 500: Internal server error

4.5 Get Hotel Bookings

Endpoint: GET /api/bookings/hotel/{hotelId}

Access Control: MANAGER, RECEPTIONIST (own hotel only), ADMIN

Path Parameters:

- hotelId: Hotel identifier (Long)

Response Body:

```
{
  "success": true,
  "message": "Hotel bookings retrieved successfully",
  "data": [
    {
      "id": 1,
      "userId": 10,
      "hotelId": 1,
      "roomId": 1,
      "roomNumber": "101",
      "roomType": "DELUXE",
      "checkInDate": "2025-01-20",
      "checkOutDate": "2025-01-25",
```

```
      "totalAmount": 750.00,
      "status": "CONFIRMED",
      "guestName": "John Doe",
      "guestEmail": "john.doe@example.com",
      "numberOfNights": 5
    },
    {
      "id": 3,
      "userId": 12,
      "hotelId": 1,
      "roomId": 5,
      "roomNumber": "105",
      "roomType": "STANDARD",
      "checkInDate": "2025-01-22",
      "checkOutDate": "2025-01-24",
      "totalAmount": 200.00,
      "status": "CHECKED_IN",
      "guestName": "Jane Smith",
      "guestEmail": "jane.smith@example.com",
      "numberOfNights": 2
    }
  ],
}
```


Status Codes:

- 200: Success
- 403: Unauthorized access
- 404: Hotel not found
- 500: Internal server error

4.6 Get All Bookings

Endpoint: GET /api/bookings

Access Control: ADMIN only

Response Body:


```
{
  "success": true,
  "message": "All bookings retrieved successfully",
```

```
  "data": [
    {
      "id": 1,
      "userId": 10,
      "hotelId": 1,
      "roomNumber": "101",
      "checkInDate": "2025-01-20",
      "checkOutDate": "2025-01-25",
      "totalAmount": 750.00,
      "status": "CONFIRMED",
      "guestName": "John Doe"
    },
    {
      "id": 2,
      "userId": 10,
      "hotelId": 2,
      "roomNumber": "205",
      "checkInDate": "2025-02-10",
      "checkOutDate": "2025-02-15",
      "totalAmount": 1250.00,
      "status": "PENDING",
      "guestName": "John Doe"
    }
  ],
}
```


Status Codes:

- 200: Success
- 403: Unauthorized access
- 500: Internal server error

4.7 Cancel Booking

Endpoint: PATCH /api/bookings/{id}/cancel

Access Control: Owner (GUEST), Hotel staff, ADMIN

Path Parameters:

- id: Booking identifier (Long)

Query Parameters:

- reason: Cancellation reason (String, optional)

Example: PATCH /api/bookings/1/cancel?reason=Change of plans

Response Body:


```
{
  "success": true,
  "message": "Booking cancelled successfully",
  "data": {
    "id": 1,
    "userId": 10,
    "hotelId": 1,
    "roomId": 1,
    "roomNumber": "101",
    "roomType": "DELUXE",
    "checkInDate": "2025-01-20",
    "checkOutDate": "2025-01-25",
    "totalAmount": 750.00,
    "status": "CANCELLED",
    "guestName": "John Doe",
    "guestEmail": "john.doe@example.com",
    "guestPhone": "+1234567890",
    "numberOfGuests": 2,
    "numberOfNights": 5,
    "specialRequests": "Late check-in requested",
    "cancellationReason": "Change of plans",
    "cancelledAt": "2025-01-15",
    "checkedInAt": null,
    "checkedOutAt": null,
    "createdAt": "2025-01-15T16:30:00",
    "updatedAt": "2025-01-15T19:00:00"
  },
}
```


Status Codes:

- 200: Booking cancelled successfully
- 400: Booking cannot be cancelled
- 403: Unauthorized access
- 404: Booking not found
- 500: Internal server error

4.8 Check-In Guest

Endpoint: POST /api/bookings/{id}/check-in

Access Control: MANAGER, RECEPTIONIST, ADMIN

Path Parameters:

- id: Booking identifier (Long)

Request Body:

```
{
  "notes": "Guest arrived 2 hours early",
  "earlyCheckIn": true
}
```

Response Body:


```
{
  "success": true,
  "message": "Guest checked in successfully",
  "data": {
    "id": 1,
    "userId": 10,
    "hotelId": 1,
    "roomId": 1,
    "roomNumber": "101",
    "roomType": "DELUXE",
    "checkInDate": "2025-01-20",
    "checkOutDate": "2025-01-25",
    "totalAmount": 750.00,
    "status": "CHECKED_IN",
    "guestName": "John Doe",
    "guestEmail": "john.doe@example.com",
    "guestPhone": "+1234567890",
    "numberOfGuests": 2,
    "numberOfNights": 5,
    "specialRequests": "Late check-in requested",
    "cancellationReason": null,
    "cancelledAt": null,
    "checkedInAt": "2025-01-20",
    "checkedOutAt": null,
    "createdAt": "2025-01-15T16:30:00",
    "updatedAt": "2025-01-20T10:00:00"
```

```
  },

}
```

Status Codes:

- 200: Check-in successful
- 400: Invalid booking status
- 403: Unauthorized access
- 404: Booking not found
- 500: Internal server error

4.9 Check-Out Guest

Endpoint: POST /api/bookings/{id}/check-out

Access Control: MANAGER, RECEPTIONIST, ADMIN

Path Parameters:

- id: Booking identifier (Long)

Request Body:

```
{
  "notes": "Guest requested late checkout",
  "rating": 5,
  "feedback": "Excellent service and comfortable stay",
  "lateCheckOut": true
}
```

Response Body:

```
{
  "success": true,
  "message": "Guest checked out successfully",
  "data": {
    "id": 1,
    "userId": 10,
    "hotelId": 1,
    "roomId": 1,
```

```
    "roomNumber": "101",
    "roomType": "DELUXE",
    "checkInDate": "2025-01-20",
    "checkOutDate": "2025-01-25",
    "totalAmount": 750.00,
    "status": "CHECKED_OUT",
    "guestName": "John Doe",
    "guestEmail": "john.doe@example.com",
    "guestPhone": "+1234567890",
    "numberOfGuests": 2,
    "numberOfNights": 5,
    "specialRequests": "Late check-in requested",
    "cancellationReason": null,
    "cancelledAt": null,
    "checkedInAt": "2025-01-20",
    "checkedOutAt": "2025-01-25",
    "createdAt": "2025-01-15T16:30:00",
    "updatedAt": "2025-01-25T12:00:00"
  },
}
```


Status Codes:

- 200: Check-out successful
- 400: Invalid booking status
- 403: Unauthorized access
- 404: Booking not found
- 500: Internal server error

4.10 Get Today's Check-Ins

Endpoint: GET /api/bookings/hotel/{hotelId}/today-checkins

Access Control: MANAGER, RECEPTIONIST, ADMIN

Path Parameters:

- hotelId: Hotel identifier (Long)

Response Body:


```
{
```

```
  "success": true,
  "message": "Today's check-ins retrieved successfully",
  "data": [
    {
      "id": 1,
      "userId": 10,
      "hotelId": 1,
      "roomId": 1,
      "roomNumber": "101",
      "roomType": "DELUXE",
      "checkInDate": "2025-01-20",
      "checkOutDate": "2025-01-25",
      "totalAmount": 750.00,
      "status": "CONFIRMED",
      "guestName": "John Doe",
      "guestEmail": "john.doe@example.com",
      "guestPhone": "+1234567890",
      "numberOfGuests": 2,
      "numberOfNights": 5
    },
    {
      "id": 5,
      "userId": 15,
      "hotelId": 1,
      "roomId": 8,
      "roomNumber": "108",
      "roomType": "SUITE",
      "checkInDate": "2025-01-20",
      "checkOutDate": "2025-01-22",
      "totalAmount": 500.00,
      "status": "CONFIRMED",
      "guestName": "Alice Johnson",
      "guestEmail": "alice.j@example.com",
      "guestPhone": "+1987654321",
      "numberOfGuests": 3,
      "numberOfNights": 2
    }
  ],
}
```


Status Codes:

- 200: Success
- 403: Unauthorized access

- 404: Hotel not found
- 500: Internal server error

4.11 Get Today's Check-Outs

Endpoint: GET /api/bookings/hotel/{hotelId}/today-checkouts

Access Control: MANAGER, RECEPTIONIST, ADMIN

Path Parameters:

- hotelId: Hotel identifier (Long)

Response Body:


```
{
  "success": true,
  "message": "Today's check-outs retrieved successfully",
  "data": [
    {
      "id": 3,
      "userId": 12,
      "hotelId": 1,
      "roomId": 5,
      "roomNumber": "105",
      "roomType": "STANDARD",
      "checkInDate": "2025-01-18",
      "checkOutDate": "2025-01-20",
      "totalAmount": 200.00,
      "status": "CHECKED_IN",
      "guestName": "Jane Smith",
      "guestEmail": "jane.smith@example.com",
      "guestPhone": "+1555666777",
      "numberOfGuests": 1,
      "numberOfNights": 2,
      "checkedInAt": "2025-01-18"
    }
  ]
}
```


Status Codes:

- 200: Success

- 403: Unauthorized access
- 404: Hotel not found
- 500: Internal server error

---

5. Data Models

5.1 Hotel Entity


```
{
  "id": 1,
  "name": "Grand Plaza Hotel",
  "description": "Luxury hotel in downtown area",
  "address": "123 Main Street",
  "city": "New York",
  "state": "NY",
  "country": "USA",
  "zipCode": "10001",
  "phone": "+1234567890",
  "email": "contact@grandplaza.com",
  "totalRooms": 150,
  "availableRooms": 145,
  "status": "ACTIVE",
  "rating": 4.5,
  "amenities": ["WiFi", "Pool", "Gym", "Restaurant"],
  "createdAt": "2025-01-15T10:30:00",
  "updatedAt": "2025-01-15T10:30:00"
}
```


5.2 Room Entity


```
{
  "id": 1,
  "hotelId": 1,
  "roomNumber": "101",
  "roomType": "DELUXE",
  "pricePerNight": 150.00,
  "status": "AVAILABLE",
  "isActive": true,
  "description": "Spacious deluxe room with city view",
```

```
  "maxOccupancy": 2,
  "amenities": ["TV", "Mini Bar", "Air Conditioning"],
  "createdAt": "2025-01-15T13:00:00",
  "updatedAt": "2025-01-15T13:00:00"
}
```

5.3 Booking Entity

```
{
  "id": 1,
  "userId": 10,
  "hotelId": 1,
  "roomId": 1,
  "roomNumber": "101",
  "roomType": "DELUXE",
  "checkInDate": "2025-01-20",
  "checkOutDate": "2025-01-25",
  "totalAmount": 750.00,
  "status": "CONFIRMED",
  "guestName": "John Doe",
  "guestEmail": "john.doe@example.com",
  "guestPhone": "+1234567890",
  "numberOfGuests": 2,
  "numberOfNights": 5,
  "cancelledAt": null,
  "checkedInAt": null,
  "checkedOutAt": null,
  "createdAt": "2025-01-15T16:30:00",
  "updatedAt": "2025-01-15T16:30:00"
}
```

---

6. Enums

6.1 Hotel Status

- ACTIVE
- INACTIVE
- MAINTENANCE
- CLOSED

6.2 Room Status

- AVAILABLE
- OCCUPIED
- MAINTENANCE

6.3 Booking Status

- PENDING
- CONFIRMED
- CHECKED_IN
- CHECKED_OUT
- CANCELLED

6.4 User Role

- ADMIN
- MANAGER
- RECEPTIONIST
- GUEST

Business rules:-

A.User & Security Rules

1. Every user must authenticate using **JWT-based authentication**
2. Role-Based Access Control (RBAC) is enforced at API level
3. Passwords are stored using strong encryption (BCrypt)
4. Users can access only permitted APIs based on role
5. Tokens are required for all protected endpoints

B.Hotel & Room Rules

1. A hotel can have multiple room categories
2. A room category defines:
   - Maximum occupancy
   - Base price

3. Each physical room belongs to:
   - One hotel
   - One room category
4. Room status must be one of:
   - Available
   - Occupied
   - Maintenance
   - Rooms under Maintenance cannot be booked

C.Pricing Rules

- **Base price is defined at room category level**

D.Reservation must include:

- Guest
- Hotel
- Room category
- Check-in date
- Check-out date

E. Reservation rules:-

1. A room **must not be double-booked** for overlapping dates
2. Reservation modification is allowed only before check-in
3. Cancelled reservations release room availability.

F. Check-in / Check-out Rules

1. Check-in:
   - Allowed only for Confirmed reservations
   - Room status becomes Occupied

2. Check-out:
   - Ends reservation
   - Room status becomes Available

3. Walk-in bookings are allowed by Receptionist only

Guest Booking flow:

Guest → Login

→ Search Rooms (Date Range)

→ View Available Rooms

→ Create Reservation

→ Reservation = Booked

→ Notification Sent

Cancellation Flow:

Booked —------> Cancelled -------> Room Availability Restored —------> Notification Sent