Hotel Booking System – API Documentation

Version: 1.0.0

Framework: FastAPI + SQLAlchemy + Motor

Last Updated: November 9, 2025

Author: Aswinnath TE

Authentication: JWT Bearer Token (`Authorization: Bearer <access_token>`)

Content Type: `application/json`

Databases: PostgreSQL (relational) + MongoDB (logs) + Redis (cache)

Overview

The **Hotel Booking System** is a comprehensive backend built with **FastAPI**. It provides role-based access for **Customers**, **Admins**, and **Super Admins**, enabling secure operations such as room management, booking lifecycle, payments, refunds, reviews, and comprehensive audit logging.

Roles & Permissions

Role	Capabilities	Access Level
Customer (role_id=1)	View rooms, make bookings, post reviews, manage wishlist, view own profile	Limited to own data
Normal Admin (role_id=3)	Manage rooms, bookings, payments, refunds, issues, view reports	Limited admin operations

Role	Capabilities	Access Level
Super Admin (role_id=2)	Full system access, user management, role/permission management, backups	Unrestricted

Permission Structure

```
Resources:
├── BOOKING (CREATE, READ, UPDATE, DELETE, MANAGE, APPROVE)
├── ADMIN_CREATION (READ, WRITE, DELETE, MANAGE, APPROVE, EXECUTE)
├── ROOM_MANAGEMENT (READ, WRITE, DELETE, MANAGE)
├── PAYMENT_PROCESSING (READ, WRITE, MANAGE)
├── REFUND_APPROVAL (READ, WRITE, DELETE, MANAGE, APPROVE)
├── CONTENT_MANAGEMENT (READ, WRITE, DELETE)
├── ISSUE_RESOLUTION (READ, WRITE, DELETE, MANAGE)
├── NOTIFICATION_HANDLING (READ, WRITE)
├── ANALYTICS_VIEW (READ)
├── BACKUP_OPERATIONS (READ, WRITE, EXECUTE)
├── RESTORE_OPERATIONS (WRITE, EXECUTE)
└── OFFER_MANAGEMENT (READ, WRITE, DELETE, MANAGE)
```

Authentication Endpoints (`/auth`)

1. Register New User (Signup)

Method	Endpoint	Auth	Description
`POST`	`/auth/signup`	No	Register a new customer account

Request Body:

```
{
  "full_name": "John Doe",
  "email": "john@example.com",
  "password": "SecurePass@123",
  "phone_number": "+1234567890",
  "dob": "1990-05-15"
}
```

Response (201 Created):

```
{
  "user_id": 42,
  "full_name": "John Doe",
  "email": "john@example.com",
  "role_id": 1,
  "created_at": "2025-11-09T10:30:00Z",
  "message": "User registered successfully"
}
```

2. Login

Method	Endpoint	Auth	Description
POST	/auth/login	No	Authenticate user and issue JWT tokens

Request Body:

```
{
  "username": "john@example.com",
  "password": "SecurePass@123"
}
```

Response (200 OK):

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "Bearer",
  "expires_in": 900,
  "role_id": 1
}
```

3. Request OTP

Method	Endpoint	Auth	Description
POST	/auth/otp/request	No	Request OTP for password reset or verification

Request Body:

```
{
  "email": "john@example.com",
  "verification_type": "PASSWORD_RESET"
}
```

Response (202 Accepted):

```
{
  "message": "OTP sent to registered email",
  "expires_in": 600,
  "otp": "123456"  // Only in development
}
```

4. Verify OTP & Reset Password

Method	Endpoint	Auth	Description
POST	/auth/otp/verify	No	Verify OTP and optionally reset password

Request Body:

```
{
  "email": "john@example.com",
  "otp": "123456",
  "verification_type": "PASSWORD_RESET",
  "new_password": "NewSecurePass@456"
}
```

Response (200 OK):

```
{
  "message": "Password reset successfully"
}
```

5. Refresh Access Token

Method	Endpoint	Auth	Yes
POST	/auth/refresh	Yes	Generate new access token using refresh token

Request Body:

```
{
  "access_token": "expired_access_token"
}
```

Response (200 OK):

```
{
  "access_token": "new_eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refresh_token": "new_refresh_token...",
  "expires_in": 900,
  "token_type": "Bearer"
}
```

6. Logout

Method	Endpoint	Auth	Yes
POST	/auth/logout	Yes	Revoke tokens and logout user

Response (200 OK):

```
{
  "message": "Logged out successfully"
}
```

7. Register Admin (Protected)

Method	Endpoint	Auth	Permission
POST	/auth/register	Yes	ADMIN_CREATION:WRITE

Request Body:

```
{
  "full_name": "Admin User",
  "email": "admin@example.com",
  "password": "SecureAdminPass@123",
  "phone_number": "+1234567890"
}
```

Response (201 Created):

```
{
  "user_id": 43,
  "full_name": "Admin User",
  "email": "admin@example.com",
  "role_id": 3,
  "message": "Admin user created successfully"
}
```

Room Management Endpoints (`/rooms`)

1. List All Rooms

Method	Endpoint	Auth	Description
`GET`	`/rooms/`	No	Fetch available rooms with filters

Query Parameters:

```
?room_type_id=2&capacity=2&price_min=50&price_max=200&limit=20&offset=0
```

Response (200 OK):


```
[
  {
    "room_id": 1,
    "room_number": "101",
    "room_type": "Deluxe",
    "capacity": 2,
    "price_per_night": 150.00,
    "status": "AVAILABLE",
    "amenities": ["WiFi", "AC", "TV"],
    "images": ["url1", "url2"],
    "created_at": "2025-01-01T00:00:00Z"
  },
  {
    "room_id": 2,
    "room_number": "102",
    "room_type": "Suite",
    "capacity": 4,
    "price_per_night": 250.00,
    "status": "AVAILABLE",
    "amenities": ["WiFi", "AC", "TV", "Gym"],
    "images": ["url1"],
    "created_at": "2025-01-01T00:00:00Z"
  }
]
```

2. Get Room Details

Method	Endpoint	Auth	Description
GET	/rooms/{room_id}	No	Get single room details

Response (200 OK):


```
{
  "room_id": 1,
  "room_number": "101",
  "room_type": "Deluxe",
  "capacity": 2,
  "price_per_night": 150.00,
  "status": "AVAILABLE",
  "amenities": ["WiFi", "AC", "TV"],
  "images": ["https://example.com/room1.jpg"],
  "description": "Comfortable deluxe room with modern amenities",
  "created_at": "2025-01-01T00:00:00Z"
}
```


3. Create Room (Admin)

Method	Endpoint	Auth	Permission
POST	/rooms/	Yes	ROOM_MANAGEMENT:WRITE

Request Body:


```
{
  "room_number": "103",
  "room_type_id": 2,
  "capacity": 3,
  "price_per_night": 200.00,
  "status_id": 1,
  "amenities": [1, 2, 3]
}
```


Response (201 Created):

```
{
  "room_id": 3,
  "room_number": "103",
  "room_type": "Deluxe Suite",
  "capacity": 3,
  "price_per_night": 200.00,
  "status": "AVAILABLE",
  "message": "Room created successfully"
}
```


4. Update Room (Admin)

Method	Endpoint	Auth	Permission
PUT	/rooms/{room_id}	Yes	ROOM_MANAGEMENT:WRITE

Request Body:


```
{
  "price_per_night": 180.00,
  "status_id": 2,
  "amenities": [1, 2, 3, 4]
}
```


Response (200 OK):


```
{
  "room_id": 1,
  "message": "Room updated successfully"
}
```


5. Delete Room (Admin)

Method	Endpoint	Auth	Permission
DELETE	/rooms/{room_id}	Yes	ROOM_MANAGEMENT:DELETE

Response (200 OK):

```
{
  "message": "Room deleted successfully"
}
```

6. Upload Room Images

Method	Endpoint	Auth	Permission
POST	/rooms/{room_id}/images	Yes	ROOM_MANAGEMENT:WRITE

Request Body: Form data with multiple files

```
Content-Type: multipart/form-data
files: [image1.jpg, image2.jpg]
```

Response (201 Created):

```
[
  {
    "image_id": 1,
    "room_id": 1,
    "image_url": "https://cdn.example.com/rooms/1/img1.jpg",
    "uploaded_at": "2025-11-09T10:30:00Z"
  },
  {
    "image_id": 2,
    "room_id": 1,
    "image_url": "https://cdn.example.com/rooms/1/img2.jpg",
    "uploaded_at": "2025-11-09T10:30:00Z"
  }
]
```

Booking Management Endpoints (`/bookings`)

1. Create Booking

Method	Endpoint	Auth	Permission
POST	/bookings/	Yes	BOOKING:WRITE

Request Body:

```
{
  "room_id": 1,
  "check_in": "2025-12-15",
  "check_out": "2025-12-18",
  "num_guests": 2,
  "special_requests": "High floor preferred"
}
```

Response (201 Created):

```
{
  "booking_id": 100,
  "user_id": 42,
  "room_id": 1,
  "check_in": "2025-12-15",
  "check_out": "2025-12-18",
  "num_guests": 2,
  "num_nights": 3,
  "status": "CONFIRMED",
  "total_price": 450.00,
  "created_at": "2025-11-09T10:30:00Z",
  "message": "Booking created successfully"
}
```

2. Get All Bookings

Method	Endpoint	Auth	Description
`GET`	`/bookings/`	Yes	Get bookings (own or all if admin)

Query Parameters:

```
?booking_id=100&status=CONFIRMED&limit=20&offset=0
```

Response (200 OK):


```
[
  {
    "booking_id": 100,
    "user_id": 42,
    "room_id": 1,
    "check_in": "2025-12-15",
    "check_out": "2025-12-18",
    "num_guests": 2,
    "status": "CONFIRMED",
    "total_price": 450.00,
    "created_at": "2025-11-09T10:30:00Z"
  },
  {
    "booking_id": 101,
    "user_id": 42,
    "room_id": 2,
    "check_in": "2025-12-20",
    "check_out": "2025-12-23",
    "num_guests": 4,
    "status": "PENDING",
    "total_price": 750.00,
    "created_at": "2025-11-08T15:00:00Z"
  }
]
```

3. Cancel Booking

Method	Endpoint	Auth	Description
POST	/bookings/{booking_id}/cancel	Yes	Cancel booking and initiate refund

Request Body:

```
{
  "reason": "Change of plans",
  "cancellation_reason": "CUSTOMER_REQUEST"
}
```

Response (201 Created):

```
{
  "refund_id": 10,
  "booking_id": 100,
  "refund_amount": 450.00,
  "refund_status": "PENDING",
  "refund_reason": "CUSTOMER_REQUEST",
  "created_at": "2025-11-09T10:30:00Z",
  "message": "Booking cancelled and refund initiated"
}
```

Payment Endpoints (`/payments`)

1. Get All Payments

Method	Endpoint	Auth	Description
GET	/payments/	Yes	Get payments (own or all if admin)

Query Parameters:

```
?payment_id=1&booking_id=100&status=SUCCESS&amount_min=100&amount_max=500&limit=20&offset=0
```

Response (200 OK):


```
[
  {
    "payment_id": 1,
    "booking_id": 100,
    "amount": 450.00,
    "status": "SUCCESS",
    "payment_method": "Credit Card",
    "gateway_id": "ch_12345abc",
    "transaction_id": "txn_123456",
    "paid_at": "2025-11-09T10:30:00Z"
  },
  {
    "payment_id": 2,
    "booking_id": 101,
    "amount": 750.00,
    "status": "PENDING",
    "payment_method": "Debit Card",
    "gateway_id": null,
    "transaction_id": null,
    "paid_at": null
  }
]
```


2. Create Payment

Method	Endpoint	Auth	Permission
POST	/payments/	Yes	PAYMENT_PROCESSING:WRITE

Request Body:


```
{
  "booking_id": 100,
  "amount": 450.00,
  "payment_method_id": 1
}
```

Response (201 Created):


```
{
  "payment_id": 1,
  "booking_id": 100,
  "amount": 450.00,
  "status": "PENDING",
  "payment_url": "https://payment-gateway.com/checkout/abc123",
  "message": "Payment initiated"
}
```


Refund Endpoints (`/refunds`)

1. Get All Refunds

Method	Endpoint	Auth	Description
`GET`	`/refunds/`	Yes	Get refunds (own or all if admin)

Response (200 OK):


```
[
  {
    "refund_id": 10,
    "booking_id": 100,
    "refund_amount": 450.00,
    "refund_status": "APPROVED",
    "refund_reason": "CUSTOMER_REQUEST",
    "approved_by": 43,
    "approved_at": "2025-11-09T11:00:00Z",
    "created_at": "2025-11-09T10:30:00Z"
  }
]
```

2. Approve Refund

Method	Endpoint	Auth	Permission
PUT	/refunds/{refund_id}/approve	Yes	REFUND_APPROVAL:APPROVE

Request Body:

```
{
  "approval_reason": "Valid cancellation request"
}
```

Response (200 OK):

```
{
  "refund_id": 10,
  "booking_id": 100,
  "refund_amount": 450.00,
  "refund_status": "APPROVED",
  "approved_by": 43,
  "approved_at": "2025-11-09T11:00:00Z",
  "message": "Refund approved successfully"
}
```

Review Endpoints (/reviews)

1. Post Review

Method	Endpoint	Auth	Description
POST	/reviews/	Yes	Post a review for a room/booking

Request Body:

```
{
  "booking_id": 100,
  "room_id": 1,
  "rating": 5,
  "comment": "Amazing experience! Clean room and excellent service.",
  "would_recommend": true
}
```

Response (201 Created):

```
{
  "review_id": 15,
  "booking_id": 100,
  "room_id": 1,
  "user_id": 42,
  "rating": 5,
  "comment": "Amazing experience! Clean room and excellent service.",
  "would_recommend": true,
  "created_at": "2025-11-09T10:30:00Z",
  "message": "Review posted successfully"
}
```

2. Get All Reviews

Method	Endpoint	Auth	Description
GET	/reviews/	No	Get reviews (filtered by room/user)

Query Parameters:

```
?room_id=1&rating_min=4&limit=20&offset=0
```

Response (200 OK):

```
[
  {
    "review_id": 15,
    "user_id": 42,
    "user_name": "John Doe",
    "room_id": 1,
    "rating": 5,
    "comment": "Amazing experience!",
    "would_recommend": true,
    "created_at": "2025-11-09T10:30:00Z"
  }
]
```


Offer Endpoints (`/offers`)

1. Get Active Offers

Method	Endpoint	Auth	Description
`GET`	`/offers/`	No	Get all active promotional offers

Response (200 OK):

```
[
  {
    "offer_id": 5,
    "code": "SAVE50",
    "description": "50% off on bookings above $200",
    "discount_percentage": 50,
    "discount_amount": null,
    "valid_from": "2025-11-01",
    "valid_to": "2025-11-30",
    "status": "ACTIVE",
    "created_at": "2025-10-15T00:00:00Z"
  },
  {
    "offer_id": 6,
    "code": "FLAT100",
    "description": "$100 flat discount",
    "discount_percentage": null,
    "discount_amount": 100.00,
    "valid_from": "2025-11-01",
    "valid_to": "2025-11-30",
    "status": "ACTIVE",
    "created_at": "2025-10-20T00:00:00Z"
  }
]
```


2. Create Offer (Admin)

Method	Endpoint	Auth	Permission
POST	/offers/	Yes	OFFER_MANAGEMENT:WRITE

Request Body:

```
{
  "code": "SAVE30",
  "description": "30% off on all bookings",
  "discount_percentage": 30,
  "valid_from": "2025-12-01",
  "valid_to": "2025-12-31"
}
```

Response (201 Created):

```
{
  "offer_id": 7,
  "code": "SAVE30",
  "discount_percentage": 30,
  "status": "ACTIVE",
  "message": "Offer created successfully"
}
```

Issue Management Endpoints (`/issues`)

1. Create Support Ticket

Method	Endpoint	Auth	Description
POST	/issues/	Yes	Submit a support ticket

Request Body:

```
{
  "title": "Room had maintenance issues",
  "description": "The AC was not working properly",
  "category": "MAINTENANCE",
  "priority": "HIGH",
  "booking_id": 100
}
```

Response (201 Created):

```
{
  "issue_id": 25,
  "user_id": 42,
  "title": "Room had maintenance issues",
  "status": "OPEN",
  "priority": "HIGH",
  "created_at": "2025-11-09T10:30:00Z",
  "message": "Issue created successfully"
}
```


2. Get All Issues

Method	Endpoint	Auth	Description
GET	/issues/	Yes	Get issues (own or all if admin)

Response (200 OK):


```
[
  {
    "issue_id": 25,
    "user_id": 42,
    "user_name": "John Doe",
    "title": "Room had maintenance issues",
    "description": "The AC was not working properly",
    "status": "OPEN",
    "priority": "HIGH",
    "created_at": "2025-11-09T10:30:00Z"
  }
]
```


3. Update Issue (Admin)

Method	Endpoint	Auth	Permission
PUT	/issues/{issue_id}	Yes	ISSUE_RESOLUTION:MANAGE

Request Body:

```
{
  "status": "RESOLVED",
  "resolution_notes": "AC unit replaced successfully"
}
```

Response (200 OK):

```
{
  "issue_id": 25,
  "status": "RESOLVED",
  "resolution_notes": "AC unit replaced successfully",
  "message": "Issue updated successfully"
}
```

Notification Endpoints (`/notifications`)

1. Get Notifications

Method	Endpoint	Auth	Description
GET	/notifications/	Yes	Get user's notifications

Query Parameters:

```
?unread_only=false&limit=20&offset=0
```

Response (200 OK):

```
[
  {
    "notification_id": 1,
    "user_id": 42,
    "title": "Booking Confirmed",
    "content": "Your booking #100 is confirmed",
    "type": "BOOKING",
    "is_read": false,
    "created_at": "2025-11-09T10:30:00Z"
  },
  {
    "notification_id": 2,
    "user_id": 42,
    "title": "Refund Approved",
    "content": "Your refund of $450 has been approved",
    "type": "REFUND",
    "is_read": true,
    "created_at": "2025-11-08T15:00:00Z"
  }
]
```


2. Mark Notification as Read

Method	Endpoint	Auth	Description
PUT	/notifications/{notification_id}/read	Yes	Mark notification as read

Response (200 OK):


```
{
  "notification_id": 1,
  "is_read": true,
  "message": "Notification marked as read"
}
```

Wishlist Endpoints (`/wishlist`)

1. Get Wishlist

Method	Endpoint	Auth	Description
`GET`	`/wishlist/`	Yes	Get user's wishlist

Response (200 OK):


```
[
  {
    "wishlist_id": 1,
    "room_id": 1,
    "room_number": "101",
    "room_type": "Deluxe",
    "price_per_night": 150.00,
    "added_at": "2025-11-01T12:00:00Z"
  },
  {
    "wishlist_id": 2,
    "room_id": 2,
    "room_number": "102",
    "room_type": "Suite",
    "price_per_night": 250.00,
    "added_at": "2025-10-28T08:30:00Z"
  }
]
```


2. Add to Wishlist

Method	Endpoint	Auth	Description
`POST`	`/wishlist/`	Yes	Add room to wishlist

Request Body:

```
{
  "room_id": 3
}
```

Response (201 Created):

```
{
  "wishlist_id": 3,
  "room_id": 3,
  "message": "Room added to wishlist"
}
```

3. Remove from Wishlist

Method	Endpoint	Auth	Description
`DELETE`	`/wishlist/{wishlist_id}`	Yes	Remove room from wishlist

Response (200 OK):

```
{
  "message": "Item removed from wishlist"
}
```

Profile Endpoints (`/profile`)

1. Get Profile

Method	Endpoint	Auth	Description
`GET`	`/profile/me`	Yes	Get current user's profile

Response (200 OK):

```
{
  "user_id": 42,
  "full_name": "John Doe",
  "email": "john@example.com",
  "phone_number": "+1234567890",
  "dob": "1990-05-15",
  "gender": "MALE",
  "role_id": 1,
  "role_name": "customer",
  "loyalty_points": 150,
  "created_at": "2025-01-01T00:00:00Z",
  "updated_at": "2025-11-09T10:30:00Z"
}
```


2. Update Profile

Method	Endpoint	Auth	Description
PUT	/profile/me	Yes	Update user's profile

Request Body:


```
{
  "full_name": "John Michael Doe",
  "phone_number": "+0987654321",
  "dob": "1990-05-15",
  "gender": "MALE"
}
```


Response (200 OK):


```
{
  "user_id": 42,
  "full_name": "John Michael Doe",
  "phone_number": "+0987654321",
  "message": "Profile updated successfully"
}
```

3. Change Password

Method	Endpoint	Auth	Description
POST	/profile/change-password	Yes	Change user's password

Request Body:

```
{
  "current_password": "SecurePass@123",
  "new_password": "NewSecurePass@456"
}
```

Response (200 OK):

```
{
  "message": "Password changed successfully"
}
```

Roles & Permissions Endpoints (/roles)

1. List All Roles

Method	Endpoint	Auth	Permission
GET	/roles/	Yes	ADMIN_CREATION:READ

Response (200 OK):

```
[
  {
    "role_id": 1,
    "role_name": "customer",
    "role_description": "Regular customer",
    "message": "Fetched successfully"
  },
  {
    "role_id": 2,
    "role_name": "super_admin",
    "role_description": "Full system access",
    "message": "Fetched successfully"
  },
  {
    "role_id": 3,
    "role_name": "normal_admin",
    "role_description": "Limited admin access",
    "message": "Fetched successfully"
  }
]
```


2. Create Role (Admin)

Method	Endpoint	Auth	Permission
POST	/roles/	Yes	ADMIN_CREATION:READ

Request Body:


```
{
  "role_name": "manager",
  "role_description": "Hotel manager with limited admin access"
}
```


Response (201 Created):

```
{
  "role_id": 4,
  "role_name": "manager",
  "role_description": "Hotel manager with limited admin access",
  "message": "Role created successfully"
}
```


3. Get Permissions

Method	Endpoint	Auth	Permission
GET	/roles/permissions	Yes	ADMIN_CREATION:READ

Query Parameters (choose one):

```
?role_id=2           # Get permissions for a role
?permission_id=5     # Get roles that have a permission
?resources=BOOKING   # Get permissions for specific resources
```

Response (200 OK):


```
{
  "role_id": 2,
  "permissions": [
    {
      "permission_id": 5,
      "resource": "BOOKING",
      "permission_type": "READ"
    },
    {
      "permission_id": 6,
      "resource": "BOOKING",
      "permission_type": "WRITE"
    }
  ]
}
```

4. Assign Permissions to Role (Admin)

Method	Endpoint	Auth	Permission
POST	/roles/assign	Yes	ADMIN_CREATION:READ

Request Body:

```
{
  "role_id": 3,
  "permission_ids": [5, 6, 11, 12]
}
```

Response (201 Created):

```
{
  "role_id": 3,
  "permissions": [
    {"permission_id": 5, "resource": "BOOKING", "permission_type": "READ"},
    {"permission_id": 6, "resource": "BOOKING", "permission_type": "WRITE"},
    {"permission_id": 11, "resource": "ADMIN_CREATION", "permission_type": "READ"},
    {"permission_id": 12, "resource": "ADMIN_CREATION", "permission_type": "WRITE"}
  ],
  "message": "Permissions assigned successfully"
}
```

Reports Endpoints (/reports)

1. Revenue Report

Method	Endpoint	Auth	Permission
GET	/reports/revenue	Yes	ANALYTICS_VIEW:READ

Query Parameters:

```
?start_date=2025-11-01&end_date=2025-11-30&group_by=daily
```

Response (200 OK):


```
{
  "report_type": "REVENUE",
  "period": {"start": "2025-11-01", "end": "2025-11-30"},
  "total_revenue": 45000.00,
  "data": [
    {
      "date": "2025-11-01",
      "daily_revenue": 1500.00,
      "bookings_count": 5
    },
    {
      "date": "2025-11-02",
      "daily_revenue": 1800.00,
      "bookings_count": 6
    }
  ]
}
```


2. Occupancy Report

Method	Endpoint	Auth	Permission
GET	/reports/occupancy	Yes	ANALYTICS_VIEW:READ

Response (200 OK):


```
{
  "report_type": "OCCUPANCY",
  "total_rooms": 50,
  "occupied_rooms": 35,
  "occupancy_rate": 70.0,
  "by_room_type": [
    {"room_type": "Deluxe", "total": 20, "occupied": 18, "rate": 90.0},
    {"room_type": "Suite", "total": 20, "occupied": 12, "rate": 60.0},
    {"room_type": "Standard", "total": 10, "occupied": 5, "rate": 50.0}
  ]
}
```

Audit & Logs Endpoints (`/audit` , `/logs`)

1. Get Audit Logs

Method	Endpoint	Auth	Permission
`GET`	`/audit/`	Yes	`ADMIN_CREATION:READ`

Query Parameters:

```
?entity=booking&entity_id=100&action=INSERT&limit=20&offset=0
```

Response (200 OK):

```
[
  {
    "audit_id": "audit_001",
    "entity": "booking",
    "entity_id": "booking:100",
    "action": "INSERT",
    "old_value": null,
    "new_value": {
      "booking_id": 100,
      "user_id": 42,
      "room_id": 1,
      "status": "CONFIRMED"
    },
    "changed_by_user_id": 42,
    "timestamp": "2025-11-09T10:30:00Z"
  },
  {
    "audit_id": "audit_002",
    "entity": "booking",
    "entity_id": "booking:100",
    "action": "UPDATE",
    "old_value": {"status": "CONFIRMED"},
    "new_value": {"status": "CANCELLED"},
    "changed_by_user_id": 42,
    "timestamp": "2025-11-09T11:00:00Z"
  }
]
```


2. Get Booking Logs

Method	Endpoint	Auth	Description
`GET`	`/logs/bookings`	Yes	Get booking transaction logs

Response (200 OK):

```
[
  {
    "log_id": "log_001",
    "booking_id": 100,
    "log_type": "CREATED",
    "timestamp": "2025-11-09T10:30:00Z",
    "details": "Booking created by user"
  }
]
```


Backup & Restore Endpoints (`/backups` , `/restores`)

1. Create Backup (Admin)

Method	Endpoint	Auth	Permission
POST	/backups/	Yes	BACKUP_OPERATIONS:WRITE

Request Body:


```
{
  "snapshot_name": "backup_2025_11_09",
  "trigger_type": "MANUAL"
}
```


Response (201 Created):


```
{
  "backup_id": "bak_001",
  "snapshot_name": "backup_2025_11_09",
  "status": "IN_PROGRESS",
  "created_at": "2025-11-09T10:30:00Z",
  "message": "Backup initiated successfully"
}
```

2. Get Backups

Method	Endpoint	Auth	Permission
GET	/backups/	Yes	BACKUP_OPERATIONS:READ

Response (200 OK):


```
[
  {
    "backup_id": "bak_001",
    "snapshot_name": "backup_2025_11_09",
    "status": "COMPLETED",
    "file_size_mb": 256.5,
    "created_at": "2025-11-09T10:30:00Z",
    "completed_at": "2025-11-09T10:45:00Z"
  }
]
```


3. Restore from Backup (Admin)

Method	Endpoint	Auth	Permission
POST	/restores/	Yes	RESTORE_OPERATIONS:WRITE

Request Body:


```
{
  "backup_id": "bak_001"
}
```


Response (201 Created):

```
{
  "restore_id": "res_001",
  "backup_id": "bak_001",
  "status": "IN_PROGRESS",
  "started_at": "2025-11-09T11:00:00Z",
  "message": "Restore process initiated"
}
```


Content Management Endpoints (`/content`)

1. Get CMS Content

Method	Endpoint	Auth	Description
`GET`	`/content/`	No	Get all published CMS content

Response (200 OK):


```
[
  {
    "content_id": 1,
    "type": "BLOG",
    "title": "10 Best Hotel Destinations",
    "description": "Discover amazing hotel destinations",
    "status": "PUBLISHED",
    "created_at": "2025-11-01T00:00:00Z"
  }
]
```


2. Create Content (Admin)

Method	Endpoint	Auth	Permission
`POST`	`/content/`	Yes	`CONTENT_MANAGEMENT:WRITE`

Request Body: Form data

```
type: BLOG
title: New Hotel Features
description: Learn about our new amenities
status: DRAFT
metadata: {"category": "updates"}
images: [file1, file2]
```

Response (201 Created):


```
{
  "content_id": 2,
  "type": "BLOG",
  "title": "New Hotel Features",
  "status": "DRAFT",
  "message": "Content created successfully"
}
```


Error Handling

Status Code	Error Type	Description
400	Bad Request	Invalid input or malformed request
401	Unauthorized	Missing or invalid JWT token
403	Forbidden	Insufficient permissions for action
404	Not Found	Requested resource does not exist
422	Unprocessable Entity	Validation error in request body
500	Internal Server Error	Unexpected server error
503	Service Unavailable	Database or external service down

Error Response Example:


```
{
  "detail": "Insufficient permissions to perform this action",
  "status_code": 403,
  "error_type": "ForbiddenError"
}
```


Token Lifecycle

Token Type	Validity	Purpose
Access Token	15 minutes	Used for every protected API call
Refresh Token	7 days	Used to renew access token
Blacklisted Token	—	Invalidated immediately upon logout

Authentication Flow:

1. User logs in → Receives access_token (15 min) & refresh_token (7 days)
2. Include access_token in Authorization header for all protected requests
3. When access_token expires → Use refresh_token at /auth/refresh endpoint
4. Get new access_token with new 15-minute window
5. On logout → Tokens are blacklisted and cannot be reused

Sample Authorization Header


```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ1c2VyXzQyIiwiZXhwIjoxNzMxM
```