Hotel Management System - API Documentation

Overview

This document provides comprehensive API documentation for the Hotel Management System microservices. The system consists of three main microservices, each exposing GraphQL endpoints for different functionalities.

Base URLs

- **Authentication Service**: `http://localhost:3001/graphql`
- **Room Management Service**: `http://localhost:3002/graphql`
- **Reservation Service**: `http://localhost:3003/graphql`

Authentication Service API

Overview

The Authentication Service handles user registration, login, and JWT token management.

GraphQL Schema

Types

```
type User {
  id: ID!
  email: String!
  firstName: String!
  lastName: String!
  role: String!
  phoneNumber: String
  address: String
  createdAt: DateTime!
  updatedAt: DateTime!
}

type AuthResponse {
```

```
  token: String!
  user: User!
}
```

Input Types

```
input RegisterInput {
  email: String!
  password: String!
  firstName: String!
  lastName: String!
  phoneNumber: String
  address: String
}

input LoginInput {
  email: String!
  password: String!
}
```

Mutations

Register User

```
mutation Register($input: RegisterInput!) {
  register(input: $input) {
    token
    user {
      id
      email
      firstName
      lastName
      role
    }
  }
}
```

Description: Creates a new user account and returns an authentication token.

Parameters: - `input`: RegisterInput object containing user details

Response: AuthResponse with JWT token and user information

Example:

```
{
  "input": {
```

```
    "email": "john.doe@example.com",
    "password": "securePassword123",
    "firstName": "John",
    "lastName": "Doe",
    "phoneNumber": "+1234567890",
    "address": "123 Main St, City, State"
  }
}
```

Login User

```
mutation Login($input: LoginInput!) {
  login(input: $input) {
    token
    user {
      id
      email
      firstName
      lastName
      role
    }
  }
}
```

Description: Authenticates a user and returns an authentication token.

Parameters: - `input`: LoginInput object with email and password

Response: AuthResponse with JWT token and user information

Queries

Get Current User

```
query Me {
  me {
    id
    email
    firstName
    lastName
    role
    phoneNumber
    address
  }
}
```

Description: Returns the current authenticated user's information.

Headers Required: - `Authorization: Bearer <JWT_TOKEN>`

Room Management Service API

Overview

The Room Management Service handles room inventory, availability, and room information management.

GraphQL Schema

Types

```
type Room {
  id: ID!
  roomNumber: String!
  roomType: String!
  price: Float!
  availability: Boolean!
  description: String
  amenities: [String!]
  maxGuests: Int!
  createdAt: DateTime!
  updatedAt: DateTime!
}
```

Input Types

```
input CreateRoomInput {
  roomNumber: String!
  roomType: String!
  price: Float!
  description: String
  amenities: [String!]
  maxGuests: Int!
}

input UpdateRoomInput {
  roomNumber: String
  roomType: String
  price: Float
  availability: Boolean
  description: String
  amenities: [String!]
```

```
  maxGuests: Int
}
```

Queries

Get All Rooms

```
query GetRooms {
  rooms {
    id
    roomNumber
    roomType
    price
    availability
    description
    amenities
    maxGuests
  }
}
```

Get Available Rooms

```
query GetAvailableRooms {
  availableRooms {
    id
    roomNumber
    roomType
    price
    availability
    description
    amenities
    maxGuests
  }
}
```

Get Room by ID

```
query GetRoom($id: ID!) {
  room(id: $id) {
    id
    roomNumber
    roomType
    price
    availability
    description
    amenities
    maxGuests
```

```
  }
}
```

Mutations

Create Room

```
mutation CreateRoom($input: CreateRoomInput!) {
  createRoom(input: $input) {
    id
    roomNumber
    roomType
    price
    availability
    description
    amenities
    maxGuests
  }
}
```

Description: Creates a new room (Admin only).

Headers Required: - `Authorization: Bearer <JWT_TOKEN>` (Admin role required)

Update Room

```
mutation UpdateRoom($id: ID!, $input: UpdateRoomInput!) {
  updateRoom(id: $id, input: $input) {
    id
    roomNumber
    roomType
    price
    availability
    description
    amenities
    maxGuests
  }
}
```

Description: Updates an existing room (Admin only).

Reservation Service API

Overview

The Reservation Service handles booking management, payment processing, and reservation lifecycle.

GraphQL Schema

Types

```graphql
type Reservation {
  id: ID!
  userId: String!
  roomId: String!
  checkInDate: DateTime!
  checkOutDate: DateTime!
  numberOfGuests: Int!
  totalPrice: Float!
  status: ReservationStatus!
  paymentId: String
  specialRequests: String
  createdAt: DateTime!
  updatedAt: DateTime!
}

enum ReservationStatus {
  PENDING
  CONFIRMED
  CANCELLED
  COMPLETED
}

type PaymentResponse {
  success: Boolean!
  paymentId: String!
  message: String!
}
```

Input Types

```graphql
input CreateReservationInput {
  userId: String!
  roomId: String!
  checkInDate: DateTime!
  checkOutDate: DateTime!
  numberOfGuests: Int!
```

```
  totalPrice: Float!
  specialRequests: String
}

input PaymentInput {
  amount: Float!
  cardNumber: String!
  expiryDate: String!
  cvv: String!
  cardholderName: String!
}
```

Queries

Get User Reservations

```
query GetUserReservations($userId: String!) {
  userReservations(userId: $userId) {
    id
    userId
    roomId
    checkInDate
    checkOutDate
    numberOfGuests
    totalPrice
    status
    paymentId
    specialRequests
  }
}
```

Get All Reservations (Admin)

```
query GetAllReservations {
  reservations {
    id
    userId
    roomId
    checkInDate
    checkOutDate
    numberOfGuests
    totalPrice
    status
    paymentId
    specialRequests
  }
}
```

Mutations

Create Reservation

```graphql
mutation CreateReservation($input: CreateReservationInput!) {
  createReservation(input: $input) {
    id
    userId
    roomId
    checkInDate
    checkOutDate
    numberOfGuests
    totalPrice
    status
    paymentId
    specialRequests
  }
}
```

Description: Creates a new reservation with automatic payment processing.

Cancel Reservation

```graphql
mutation CancelReservation($id: ID!) {
  cancelReservation(id: $id) {
    id
    status
  }
}
```

Description: Cancels an existing reservation.

Process Payment (Mock)

```graphql
mutation ProcessPayment($input: PaymentInput!) {
  processPayment(input: $input) {
    success
    paymentId
    message
  }
}
```

Description: Processes payment using the mock payment gateway.

Error Handling

All APIs return standard GraphQL error responses:

```
{
  "errors": [
    {
      "message": "Error description",
      "locations": [{"line": 2, "column": 3}],
      "path": ["fieldName"]
    }
  ]
}
```

Authentication

Most endpoints require JWT authentication. Include the token in the Authorization header:

```
Authorization: Bearer <JWT_TOKEN>
```

Rate Limiting

- Authentication endpoints: 5 requests per minute per IP
- Other endpoints: 100 requests per minute per user

Status Codes

- `200`: Success
- `400`: Bad Request
- `401`: Unauthorized
- `403`: Forbidden
- `404`: Not Found
- `500`: Internal Server Error