API List & System Design for Guest House Booking (Node.js + MySQL)

# 1. Authentication & Token Logic

All protected APIs will require a valid JWT token passed in the Authorization header as:  
Authorization: Bearer <token>  
- Token is generated upon login.  
- Used to authenticate and authorize access to protected routes.

# 2. Node.js API List with Descriptions

## 2.1 Authentication APIs

- POST /api/auth/login: Login for Admin & Staff using email/phone and password. Returns JWT Token.  
  
- POST /api/auth/verify-otp: Verify OTP for user to access booking history.  
  
- POST /api/auth/logout: Invalidate JWT token (optional).

## 2.2 Public Frontend APIs (No Auth Required)

- GET /api/rooms/types: Get room types (Deluxe/Suite) with pricing.  
  
- POST /api/rooms/availability: Check room availability by date range.  
  
- POST /api/bookings/check: User checks past bookings via phone and OTP.  
  
- GET /api/rooms/facilities: List available room facilities.

## 2.3 Booking APIs (User)

- POST /api/bookings/create: Create booking with user details, membership info, and ID proof.  
  
- POST /api/payments/initiate: Initiate online/card payment.  
  
- POST /api/payments/verify: Verify payment response.  
  
- GET /api/bookings/history: Get user’s booking history (requires OTP token).

## 2.4 Admin Panel APIs (JWT Required)

- POST /api/admin/rooms/create: Add new room.  
  
- PUT /api/admin/rooms/update/:id: Update room details.  
  
- DELETE /api/admin/rooms/delete/:id: Delete a room.  
  
- POST /api/admin/rooms/set-price: Set room pricing based on date and member status.  
  
- POST /api/admin/rooms/upload-image: Upload photos for room.  
  
- POST /api/admin/bookings/create-walkin: Book for walk-in customer.  
  
- GET /api/admin/customers/history: View booking history.  
  
- POST /api/admin/users/create-staff: Create staff user.  
  
- GET /api/admin/users/list: List all staff users.

## 2.5 Reports APIs (JWT Required)

- GET /api/reports/revenue: Get revenue reports with date filters.  
  
- GET /api/reports/frequent-customers: List users with multiple bookings.

# 3. MySQL Database Structure (Basic Overview)

* Table: users

1. - id
2. - name
3. - phone
4. - email
5. - password\_hash
6. - user\_type (admin/staff/customer)
7. - created\_at

* Table: rooms

1. - id
2. - room\_type (Deluxe/Suite)
3. - room\_number
4. - status (available/booked)
5. - facilities
6. - created\_at

* Table: room\_prices

1. - id
2. - room\_type
3. - date\_from
4. - date\_to
5. - price\_member
6. - price\_non\_member

* Table: bookings

1. - id
2. - user\_id
3. - room\_id
4. - from\_date
5. - to\_date
6. - is\_member
7. - membership\_number
8. - id\_proof\_url
9. - payment\_status
10. - payment\_mode
11. - created\_at

* Table: payments

1. - id
2. - booking\_id
3. - amount
4. - mode
5. - status
6. - transaction\_id
7. - created\_at

* Table: staff\_users

1. - id
2. - name
3. - email
4. - password\_hash
5. - role
6. - created\_at

* Table: room\_photos

1. - id
2. - room\_id
3. - image\_url

* Table: reports\_cache (optional)

1. - id
2. - data
3. - created\_at