Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the backend architecture for **RoomEase - Hotel Booking Backend**. The system provides RESTful APIs for user registration, authentication, and hotel room booking management using Node.js, Express, and MongoDB.

---

1. System Components

Component	Description	Key Responsibilities
API Server	Express.js application	Route requests, error handling
Auth Module	JWT-based authentication	User login, token validation
User Service	User management logic	Register, authenticate users
Booking Service	Booking management logic	Create, view, cancel bookings
Room Service	Room inventory management	List, update room availability
MongoDB Database	Data persistence	Store users, rooms, bookings

---

2. Class/Interface Overview

Class/Interface	Description	Key Methods/Attributes
UserController	Handles user endpoints	`register()`, `login()`
BookingController	Handles booking endpoints	`createBooking()`, `getBookings()`, `cancelBooking()`
RoomController	Handles room endpoints	`listRooms()`, `updateRoomStatus()`
AuthMiddleware	JWT validation	`authenticate()`
UserModel	User schema/model	`username`, `email`, `passwordHash`
RoomModel	Room schema/model	`roomNumber`, `type`, `status`
BookingModel	Booking schema/model	`userId`, `roomId`, `startDate`, `endDate`, `status`

---

3. Data Structure Overview

Model	Fields
User	`_id`, `username`, `email`, `passwordHash`, `createdAt`
Room	`_id`, `roomNumber`, `type`, `status` (`available` / `booked`), `price`

Booking	`_id`, `userId`, `roomId`, `startDate`, `endDate`, `status` (`active` / `cancelled`), `createdAt`

---

4. Algorithms/Logic

Booking Creation Flow (Pseudocode):

```
function createBooking(userId, roomId, startDate, endDate) {
  if (!isRoomAvailable(roomId, startDate, endDate)) {
    throw Error('Room not available');
  }
  booking = new Booking({ userId, roomId, startDate, endDate, status: 'active' });
  updateRoomStatus(roomId, 'booked');
  save(booking);
  return booking;
}
```

Authentication Flow (Pseudocode):

```
function login(email, password) {
  user = findUserByEmail(email);
  if (!user || !verifyPassword(password, user.passwordHash)) {
    throw Error('Invalid credentials');
  }
  token = generateJWT(user._id);
  return token;
}
```

---

5. Error Handling

Scenario	Handling Approach
Invalid credentials	Return 401 Unauthorized
Room not available	Return 409 Conflict
Invalid/missing JWT	Return 401 Unauthorized
Resource not found (user/room)	Return 404 Not Found
Validation errors (input)	Return 400 Bad Request
Database/server errors	Return 500 Internal Server Error

End of Document