High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **RoomEase - Hotel Booking Backend**. The system provides backend APIs for hotel room bookings, user registration, authentication, and booking management, leveraging Node.js, Express, and MongoDB.

---

1. System Architecture Overview

Architecture Description:
RoomEase follows a modular, RESTful backend architecture. The system consists of an API server built with Node.js and Express, interfacing with a MongoDB database. Authentication is handled via JWT. The backend exposes endpoints for user and booking management.

Module	Description
API Server	Handles HTTP requests, routes, and responses
Authentication	Manages user registration, login, JWT issuance
User Management	CRUD operations for user profiles
Booking Management	CRUD operations for hotel room bookings
Database Layer	MongoDB collections for users and bookings

---

2. Component Interactions

Interaction Step	Description
1. Client → API Server	Sends HTTP requests (register, login, booking)
2. API Server → Authentication Module	Validates credentials, issues JWT
3. API Server → User/Booking Management Modules	Processes user/booking requests
4. Management Modules → Database Layer	Reads/writes user and booking data
5. API Server → Client	Returns responses (data, status, errors)

---

3. Data Flow Overview

Data Flow	Source	Destination	Purpose
User Registration/Login	Client	API Server	Create/authenticate user
JWT Token Issuance	Auth Module	Client	Secure session management
Booking Creation/Update	Client	API Server	Manage room bookings

Data Persistence	API Server	MongoDB	Store/retrieve user/bookings
Booking/User Data Retrieval	API Server	Client	Provide booking/user info

---

4. Technology Stack

Layer/Component	Technology/Framework
Language/Runtime	Node.js
Web Framework	Express.js
Database	MongoDB
Authentication	JWT (JSON Web Tokens)
API Protocol	REST
Data Modeling	Mongoose (ODM)

---

5. Scalability & Reliability

- **Scalability:**
  Stateless API design enables horizontal scaling. MongoDB supports sharding for large datasets.
- **Reliability:**
  Input validation, error handling, and JWT-based authentication ensure secure and reliable operations.
- **Security:**
  Passwords are hashed; JWT tokens are used for session management; endpoints are protected via middleware.

End of Document