Product Requirements & Specification Document

Project Name

RoomEase - Hotel Booking Backend

Description

RoomEase is a backend API for managing hotel room bookings. It supports user registration, authentication, and booking management. The system is built using Node.js, Express, and MongoDB, targeting startup environments.

---

1. Goals & Objectives

Goal	Description
User Management	Register, authenticate, and manage users
Room Management	CRUD operations for hotel rooms
Booking Management	Create, view, and cancel room bookings
Secure API	Ensure secure access via authentication
Scalable Architecture	Modular, maintainable, and scalable backend

---

2. Functional Requirements

2.1 User Management

Feature	Description
Registration	Users can register with email & password
Login	Users authenticate and receive JWT token
Profile	Users can view and update their profile

2.2 Room Management

Feature	Description
Add Room	Admins can add new rooms
Edit Room	Admins can update room details
Delete Room	Admins can remove rooms
List Rooms	Users can view available rooms

2.3 Booking Management

Feature	Description
Create Booking	Users can book available rooms
View Bookings	Users can view their bookings
Cancel Booking	Users can cancel their bookings
Admin View	Admins can view all bookings

---

3. Non-Functional Requirements

Requirement	Specification
Security	JWT authentication, password hashing
Performance	API response < 500ms for standard operations
Scalability	Modular codebase, stateless API
Documentation	API documented via Swagger/OpenAPI
Error Handling	Consistent error responses (JSON)

---

4. API Endpoints

4.1 Authentication

Method	Endpoint	Description	Auth Required
POST	/api/register	Register user	No
POST	/api/login	Login user	No

4.2 User

Method	Endpoint	Description	Auth Required
GET	/api/profile	Get user profile	Yes
PUT	/api/profile	Update user profile	Yes

4.3 Rooms

Method	Endpoint	Description	Auth Required	Admin Only
GET	/api/rooms	List rooms	No	No
POST	/api/rooms	Add room	Yes	Yes
PUT	/api/rooms/:id	Edit room	Yes	Yes
DELETE	/api/rooms/:id	Delete room	Yes	Yes

4.4 Bookings

Method	Endpoint	Description	Auth Required	Admin Only

POST	/api/bookings	Create booking	Yes	No
GET	/api/bookings	List user bookings	Yes	No
DELETE	/api/bookings/:id	Cancel booking	Yes	No
GET	/api/admin/bookings	List all bookings	Yes	Yes

---

5. Data Models (MongoDB)

User

```
{
  _id: ObjectId,
  name: String,
  email: String,
  password: String (hashed),
  role: String ('user' | 'admin'),
  createdAt: Date
}
```

Room

```
{
  _id: ObjectId,
  name: String,
  description: String,
  price: Number,
  capacity: Number,
  amenities: [String],
  available: Boolean,
  createdAt: Date
}
```

Booking

```
{
  _id: ObjectId,
  userId: ObjectId,
  roomId: ObjectId,
  startDate: Date,
  endDate: Date,
  status: String ('active' | 'cancelled'),
  createdAt: Date
}
```

---

6. Security & Validation

- Passwords hashed (bcrypt)
- JWT for authentication
- Input validation (Joi or similar)
- Role-based access for admin endpoints

---

7. Technology Stack

Component	Technology
Language	Node.js
Framework	Express
Database	MongoDB
Auth	JWT
Validation	Joi (or similar)
Docs	Swagger/OpenAPI

---

8. Milestones

Milestone	Description
User Auth	Registration & login endpoints
Room CRUD	Room management endpoints
Booking Management	Booking endpoints
Admin Features	Admin-only endpoints
API Documentation	Swagger/OpenAPI docs
Testing & Deployment	Unit tests, deployment scripts

---

9. Out of Scope

- Payment processing
- Frontend/UI
- Multi-hotel support
- Advanced analytics

---

10. Appendix

Example Booking Flow (Pseudocode)

```
// User books a room
POST /api/bookings
```

```
{
  "roomId": "roomObjectId",
  "startDate": "YYYY-MM-DD",
  "endDate": "YYYY-MM-DD"
}
```

End of Document