📑 Rental Application Documentation

1. Introduction

This document describes the **Rental Management Application** that connects **Tenants, Owners, and Admins** with a **payment system**.
The app manages properties, rooms, leases, rent payments, owner payouts, and disputes.

---

2. Roles & Responsibilities

2.1 Tenant

- Register & manage profile
- Book room/bed
- Pay rent (manual / auto-pay)
- Raise disputes if needed

2.2 Owner

- Add properties, rooms, and beds
- Manage leases (approve/reject tenants)
- Receive payouts

2.3 Admin

- Approve owners/properties
- Monitor all transactions
- Resolve disputes
- Generate system reports

---

3. Core Modules

3.1 User Management

- Users table stores all users (tenant, owner, admin)
- Roles assigned per user
- Authentication & authorization

3.2 Property & Room Management

- Owners create properties → rooms → beds
- Status: available / occupied

3.3 Lease Management

- Lease = Agreement between Tenant & Owner

- Fields: start date, end date, rent amount, security deposit, billing cycle
- Only one active lease per bed/room

3.4 Payments

- Tenants pay rent via UPI, Card, or Bank
- Transactions recorded in DB
- Auto-pay option available
- Status: pending | success | failed

3.5 Payouts

- Money held in system → released to owner
- Gateway fees deducted
- Payouts tracked with status

3.6 Disputes

- Tenant/Owner can raise disputes
- Admin reviews & resolves
- Status updated accordingly

---

4. Database Schema (Main Tables)

1. **Users** → Tenant / Owner / Admin details
2. **Properties** → Property info
3. **Rooms** → Rooms inside properties
4. **Beds** → Beds inside rooms (optional, PG style)
5. **Leases** → Agreements
6. **Payment Methods** → UPI / Card / Bank
7. **Transactions** → Rent payments
8. **Payouts** → Owner settlements
9. **Disputes** → Issue tracking

(SQL queries we already prepared, can be attached in Annexure)

---

5. Business Logic

5.1 Tenant Flow

1. Register → Search Property → Book Room/Bed

2. Lease created → Start date, Rent set
3. Pay rent (manual/auto-pay)
4. Transaction stored

5.2 Owner Flow

1. Register → Add Property → Add Rooms/Beds
2. Approve tenants
3. Receive payouts (after tenant pays)

5.3 Admin Flow

1. Approve owners/properties
2. Monitor payments & payouts
3. Resolve disputes
4. Generate reports

---

6. Payment Workflow

1. Tenant initiates rent payment
2. Payment Gateway processes transaction
3. If success → Transactions entry created
4. System deducts fee → Owner payout queued
5. Owner receives money → Payout entry completed

---

7. Dispute Workflow

1. Tenant/Owner raises dispute (wrong payment / double deduction)
2. Dispute entry created
3. Admin investigates → Updates status (open | resolved | rejected)
4. Refund or adjustment made if valid

---

8. Technology Stack (suggested)

- **Frontend:** React.js / Flutter
- **Backend:** Node.js (Express) / Python (Django/FastAPI) / Java (Spring Boot)
- **Database:** PostgreSQL
- **Payments:** Razorpay / Stripe / Paytm UPI API

- **Hosting:** AWS / GCP / Azure

---

9. Security Considerations

- Encrypt sensitive user data
- PCI DSS compliance for payment storage
- Role-based access control
- Logs for disputes & payouts

---

10. Future Enhancements

- Tenant credit score system
- Owner rating & reviews
- Chat between tenant & owner
- Dynamic rent reminders & push notifications