Rent Today Web Application Documentation

1. Project Overview

- **Project Name**: Rent Today
- **Objective**: To create a platform that connects landlords with customers seeking rental properties.
- **Key Features**:
  - Property listing and search
  - Landlord and customer registration/login
  - Property details with images, pricing, and descriptions
  - Booking requests and management
  - Customer reviews and ratings
  - Admin dashboard for managing listings and users

2. Project Goals

- Simplify the rental process by providing an easy-to-use web platform.
- Enable landlords to list properties and manage bookings.
- Allow customers to search for properties based on filters (e.g., location, price, type).

3. Technology Stack

- **Frontend**: React.js
- **Backend**: Node.js with Express.js
- **Database**: PostgreSQL
- **Authentication**: JSON Web Tokens (JWT)
- **Hosting**:
  - Backend: Render
  - Frontend: Netlify or Vercel
  - Database: Neon

4. Application Features

4.1 User Roles

- **Landlord**:
  - Create and manage property listings
  - Accept or reject booking requests
  - View customer feedback
- **Customer**:
  - Search and filter properties
  - Book properties
  - Leave reviews and ratings
- **Admin**:
  - Manage users (landlords/customers)
  - Monitor property listings and bookings

4.2 Functionalities

- **Authentication**: User registration, login, and role-based access.
- **Search and Filters**: Location-based search, price range, and property type.
- **Property Management**: Upload images, descriptions, and availability status.
- **Notifications**: Booking confirmation and updates via email or in-app messages.

5. System Architecture

• **Frontend**:

- Design and implement the header, tabs, property cards, and WhatsApp button.
- Fetch dynamic data from the backend using Axios or Fetch API.

• **Backend**:

- Create API routes to fetch data for "Picked for You" and "Recent Viewed."
- Add logic to track and update property views in the database.

• **Testing**:

- Ensure the layout works across devices.
- Test API integrations for dynamic data loading.

- **Database**:
  - Tables:
    - Users (id, name, email, role, password)
    - Properties (id, landlord_id, title, description, price, location, images)
    - Bookings (id, customer_id, property_id, status, date)
- **Database Schema**
- **Users Table**

Field	Type	Description
id	UUID	Unique identifier
name	VARCHAR	Full name
email	VARCHAR	User email (unique)
password	VARCHAR	Hashed password
role	ENUM	'customer', 'landlord'

- **Properties Table**

Field	Type	Description
id	UUID	Unique identifier
landlord_id	UUID	Linked to the landlord (user id)
title	VARCHAR	Title of the property
description	TEXT	Full description
price	FLOAT	Monthly rent
bedrooms	INT	Number of bedrooms

Field	Type	Description
location	VARCHAR	Address or area
images	JSON	Array of image URLs

- **Bookings Table**

Field	Type	Description
id	UUID	Unique identifier
customer_id	UUID	Linked to the customer (user id)
property_id	UUID	Linked to the property
status	ENUM	'pending', 'accepted', 'rejected'
date	TIMESTAMP	Booking date

6. Development Workflow

- **Version Control**: Git with GitHub
- **Project Management**: ClickUp,
- **CI/CD**: Github actions for automatic builds and deployments.