1. Ruang Lingkup Kerja (Scope)

- **Tenant = Nagari**
    - Setiap nagari adalah tenant.
    - Akses bisa melalui **subdomain** (`lubukbasung.appmu.com`) atau **custom domain** (`nagari-lubukbasung.id`).
    - Semua nagari berada di **satu database**, dibedakan dengan `tenant_id`.
- **User & Roles**
    - **Staf Nagari** → terikat pada satu nagari, hanya bisa mengakses data nagari-nya sendiri.
    - **Staf Global (Induk)** → memiliki flag `is_global = true`, bisa mengakses semua nagari tanpa batas.
    - **Role** → level peran (Admin, Staff, Viewer).
    - **Sub-role** (opsional) → granular permission di bawah role, misalnya Admin → Bendahara, Sekretaris, Operator.
- **Data Per Nagari**
    - Semua data (contoh: warga, surat, arsip, dll) harus memiliki `tenant_id`.
    - User hanya bisa mengakses data dengan `tenant_id` sesuai dengan nagari yang sedang diakses (kecuali staf global).
- **Akses Sistem**
    - Middleware `TenantMiddleware` → resolve tenant berdasarkan domain/subdomain.
    - Middleware `CheckRole` → cek apakah user memiliki role sesuai tenant aktif.
    - **Staf Global** → bypass semua tenant check.

2. Use Case

A. Global Admin (Staf Induk)

- Login via `appmu.com`.
- Melihat daftar semua nagari.
- Mengakses data lintas nagari (laporan agregat, monitoring).
- Mengelola user global maupun user nagari.

B. Admin Nagari

- Login via subdomain (`lubukbasung.appmu.com`) atau domain (`nagari-lubukbasung.id`).
- Mengelola staf nagarinya.
- Melihat & mengelola data warga yang hanya terikat pada nagarinya (`tenant_id`).

C. Staff Nagari

- Login via domain/subdomain nagari.
- Hanya mengakses menu sesuai role & sub-role.
    - Contoh: Staff Keuangan → hanya bisa akses menu laporan keuangan.
    - Contoh: Staff Operator → hanya bisa input data warga.

D. Warga (opsional, jika ditambahkan)

- Warga bisa login untuk mengajukan surat online.
- Data otomatis terkait dengan `tenant_id` sesuai nagari tempatnya terdaftar.

3. Desain Database

Tabel Utama

```
tenants (nagaris)
- id
- name
- domain
- subdomain
- type (domain/subdomain)
- created_at
- updated_at

users
- id
- name
- email
- password
- is_global (boolean)
- created_at
- updated_at

roles
- id
- name
- description
- created_at
- updated_at

sub_roles
- id
- role_id (FK -> roles.id)
- name
- description
- created_at
- updated_at
```

Pivot Tables

```
role_user
- id
```

```
- user_id (FK -> users.id)
- role_id (FK -> roles.id)
- tenant_id (FK -> tenants.id, nullable)
- created_at
- updated_at

sub_role_user
- id
- user_id (FK -> users.id)
- sub_role_id (FK -> sub_roles.id)
- tenant_id (FK -> tenants.id, nullable)
- created_at
- updated_at
```

Contoh Data Tambahan

```
wargas
- id
- tenant_id (FK -> tenants.id)
- nama
- nik
- alamat
- created_at
```

4. Relasi Antar Tabel

- **Nagari** ↔ **Users** (Many-to-Many via `role_user`)
- **User** ↔ **Role** (Many-to-Many via `role_user`)
- **Role** ↔ **SubRole** (One-to-Many)
- **User** ↔ **SubRole** (Many-to-Many via `sub_role_user`)
- **Nagari** ↔ **Data (Warga, Surat, Arsip)** (One-to-Many)

Diagram sederhananya:

```
Nagari (Tenant) ---< role_user >--- Users
                 |                   |
                 |                   |
                 v                   v
              Roles ---< sub_roles >--- sub_role_user --- Users
```

---

5. Summary

- Semua data **terikat ke nagari** via `tenant_id`.
- **Staf Global (is_global = true)** → akses semua nagari.
- **Staf Nagari** → hanya akses data sesuai nagari aktif (domain/subdomain).
- Middleware akan **resolve tenant** dan **filter akses** otomatis.
- Arsitektur ini fleksibel untuk subdomain maupun domain custom.