Invento
SOFTWARE LIMITED

Multi-Tenant Flat & Bill Management System

Scenario

You are tasked with building a **multi-tenant system** to manage buildings, flats, and bills. The system should support multiple house owners (each owning a building) and allow Admins to manage tenants across all buildings.

Entities:

1. **Admin (Super Admin)** – manages the whole system.
2. **House Owner** – owns a building with multiple flats.

Important:

- The system must be **multi-tenant**, ensuring **data isolation**: House Owners can only access their own building and flats.

User Roles & Permissions Admin (Super Admin)

1. Can create and manage **House Owners**.
2. Can create **Tenants**.
3. Can view **Tenant details**.
4. Can remove tenants.
5. Can assign tenants to buildings.

House Owner

1. Can create **Flats** in their building.
2. Can manage flat details (flat number, flat owner details).
3. Can create **Bill Categories**:

- Electricity
- Gas bill
- Water bill
- Utility Charges

4. Can create **Bills** for flats.
5. Can add **due amounts** if a flat hasn’t paid previous bills.
6. Receives **email notifications** when:
   - A new bill is created.
   - A bill is paid.

Functional Requirements

1. **Multi-Tenant Isolation**
   - House Owners cannot see other owners’ buildings, flats, tenants, or bills.
2. **Flats Management**
   - Flats must include flat number and owner details.
   - House Owner can create, update, and delete flats.
3. **Tenant Management**
   - Tenants are assigned by Admin to a building (House Owner).
   - Tenant details include name, contact, and email.
4. **Bill Management**
   - Bills are assigned to flats.
   - Include: month, bill type (category), amount, status (paid/unpaid), and optional notes.

    - Dues management: if a bill is unpaid, the due can be carried forward.
    - Email notifications:
        - Bill created → notify relevant stakeholders.
        - Bill paid → notify House Owner/Admin.

Technical Requirements

1. **Tech Stack:**
    - Laravel (any recent version)
    - Frontend: Bootstrap or Tailwind CSS (UI is not the focus)
    - MySQL or PostgreSQL database
2. **Performance:**
    - Queries must be optimized
    - Code must be clean, reuseable, and maintainable.
3. **Multi-Tenant Approach:**
    - Subdomain or column-based tenant identification is acceptable.
    - Data isolation must be enforced at the query and middleware level.
4. **Email Notifications:**
    - On bill creation and bill payment.
5. **Documentation:**
    - Provide proper README with setup instructions.
    - Include SQL file to create the database and seed sample data.

Deliverables

1. Fully working Laravel project (migrations, seeders, models, controllers).
2. SQL file for database structure and sample data.
3. README.md including:
   - Setup instructions (local development, subdomains if used).
   - Short description of multi-tenant implementation.
   - Notes on optimization, queries, and design decisions.

Notes for Candidate

1. **UI is not the focus** — simple Bootstrap/Tailwind forms and tables are sufficient.
2. **Code and queries must be optimized** — clean, maintainable code and efficient database queries are required.
3. **Proper Documentation is required** — include database structure, sample data, and clear setup instructions.
4. **Email notifications** should be functional — local mail testing is acceptable.
5. **Use of AI tools is prohibited** — the task must be completed manually.
6. **Timeframe** — the task must be completed within **2 days**.
7. **Submission** — the project must be submitted via **email** with a **public Git repository link**.