PG Management System (Django)

1. Project Overview

The **PG Management System** is a Django-based web application designed to manage Paying Guest (PG) accommodations. It provides administrative capabilities for managing rooms, facilities, tenants, electricity bills, and master data through the Django Admin panel and REST APIs.

---

2. Technology Stack

- **Backend Framework**: Django, Django REST Framework (DRF)
- **Database**: SQLite (development), scalable to PostgreSQL/MySQL
- **Admin Interface**: Django Admin
- **API Layer**: REST APIs using DRF
- **Environment**: Python 3.x, Virtual Environment

---

3. Project Structure

```
pg_management/
│
├── pg_managementPRD/
│   ├── settings.py
│   ├── urls.py
│   ├── asgi.py
│   ├── wsgi.py
│   └── __init__.py
│
├── MASTERAPP/
│   ├── migrations/
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
│   ├── tests.py
│   └── __init__.py
│
├── tenant_docs/
├── db.sqlite3
├── manage.py
```

```
├── requirements.txt
└── venv/
```

---

4. Core Applications

4.1 MASTERAPP

This is the main application containing all business logic and database models.

Handled modules: - Room Management - Facility Management - Tenant Management - Electricity Billing - Master Data (Access Types, Login Types, User Types)

---

5. Database Models

5.1 BaseModel (Abstract)

Common fields shared across all models.

Typical Fields: - `id` - `created_at` - `updated_at` - `is_active`

5.2 Facility Model

Purpose: Stores facilities available in the PG.

Fields: - `facility_name` (CharField) - `sr_no` (PositiveIntegerField, Unique) - `remarks` (TextField)

Database Table: `Facility`

5.3 Room Model

Purpose: Manages room details and configurations.

Choice Fields:

- **Room Type**:
- Single
- Double
- Triple

- Dormitory

- **AC Type**:

- AC
- Non-AC

Fields: - `room_type` (CharField) - `ac_non_ac` (CharField) - `room_rent` (IntegerField) - `room_number` (CharField / IntegerField) - `remarks` (TextField, Optional)

---

6. Django Admin Panel

6.1 Features

- Add, update, delete rooms
- Filter rooms by:
- Room Type
- AC / Non-AC
- Bulk delete support
- Search functionality

6.2 Admin Configuration

Each model is registered in `admin.py` with: - `list_display` - `list_filter` - `search_fields`

---

7. Serializers (DRF)

Serializers are used to convert model instances to JSON and vice versa.

Location: `MASTERAPP/serializers.py`

Used for: - API responses - Input validation

---

8. Views & APIs

Location: `MASTERAPP/views.py`

API Capabilities

- Create Room
- List Rooms
- Update Room
- Delete Room

Uses: - `APIView` or `ModelViewSet` - Token / Session Authentication (configurable)

9. URL Configuration

Project URLs

Defined in:

```
pg_managementPRD/urls.py
```

App URLs

Each app API is routed and included in the main URLs file.

---

10. Authentication & Authorization

- Django built-in authentication
- Admin-only access for management modules
- Permissions handled via Django Admin & DRF permissions

---

11. Environment Setup

11.1 Create Virtual Environment

```
python -m venv venv
```

11.2 Activate Virtual Environment

```
# Windows
venv\Scripts\activate

# Linux / Mac
source venv/bin/activate
```

11.3 Install Dependencies

```
pip install -r requirements.txt
```

---

12. Database Migration

```
python manage.py makemigrations
python manage.py migrate
```

---

13. Create Superuser

```
python manage.py createsuperuser
```

---

14. Run Development Server

```
python manage.py runserver
```

Admin Panel:

```
http://127.0.0.1:8000/admin/
```

---

15. Best Practices Followed

- Modular app structure
- Use of choices for controlled fields
- Clean admin filters
- Reusable base model
- Separation of concerns (models, serializers, views)

---

16. Future Enhancements

- Tenant allocation to rooms
- Payment & invoice module
- Role-based access control
- Production database support (PostgreSQL)
- API documentation (Swagger / Redoc)

---

17. Conclusion

This PG Management System provides a scalable and maintainable backend foundation for managing PG operations efficiently using Django and Django REST Framework.