Django Assignment: Building a Healthcare Backend

Objective:

The goal of this assignment is to create a backend system for a healthcare application using Django, Django REST Framework (DRF), and PostgreSQL. The system should allow users to register, log in, and manage patient and doctor records securely.

Requirements:

- Use **Django** and **Django REST Framework (DRF)** for the backend.
- Use **PostgreSQL** as the database.
- Implement **JWT authentication** for user security using **djangorestframework-simplejwt**.
- Create RESTful API endpoints for managing patients and doctors.
- Use Django ORM for database modeling.
- Implement **error handling and validation**.
- Use **environment variables** for sensitive configurations.

APIs to be Implemented:

1. Authentication APIs

- **POST** `/api/auth/register/` - Register a new user with name, email, and password.
- **POST** `/api/auth/login/` - Log in a user and return a JWT token.

2. Patient Management APIs

- **POST** `/api/patients/` - Add a new patient (Authenticated users only).
- **GET** `/api/patients/` - Retrieve all patients created by the authenticated user.
- **GET** `/api/patients/<id>/` - Get details of a specific patient.
- **PUT** `/api/patients/<id>/` - Update patient details.
- **DELETE** `/api/patients/<id>/` - Delete a patient record.

3. Doctor Management APIs

- **POST** `/api/doctors/` - Add a new doctor (Authenticated users only).
- **GET** `/api/doctors/` - Retrieve all doctors.
- **GET** `/api/doctors/<id>/` - Get details of a specific doctor.
- **PUT** `/api/doctors/<id>/` - Update doctor details.
- **DELETE** `/api/doctors/<id>/` - Delete a doctor record.

4. Patient-Doctor Mapping APIs

- **POST** `/api/mappings/` - Assign a doctor to a patient.
- **GET** `/api/mappings/` - Retrieve all patient-doctor mappings.
- **GET** `/api/mappings/<patient_id>/` - Get all doctors assigned to a specific patient.
- **DELETE** `/api/mappings/<id>/` - Remove a doctor from a patient.

Instructions:

1. Set up a Django project with Django REST Framework and PostgreSQL.
2. Use Django ORM for database interaction.
3. Implement authentication using JWT with `djangorestframework-simplejwt`.
4. Secure patient and doctor-related endpoints with authentication permissions.
5. Follow best practices for structuring the project.
6. Test all API endpoints using Postman or an API client.

Expected Outcome:

- Users should be able to register and log in.
- Authenticated users should be able to add and manage patient and doctor records.
- Patients should be able to be assigned to doctors.
- Data should be stored securely in PostgreSQL.

Good luck! 🚀