DTxPlus

Dental Management Platform

Naitik Rathod - Full Stack Developer Application

GitHub URL:
https://github.com/NaitikR/dental_management_system.git

Introduction

This project is a Dental Management Platform designed for managing clinics, doctors, and patients. It provides administrative users with tools to schedule appointments and track visits.

Features

User Authentication: Secure login system using Django's built-in authentication.

Clinics Management: Add, view, and edit clinics with doctor affiliations.

Doctors Management: Manage doctors with specialties and clinic affiliations.

Patient Management: Handles patient information, visits, and appointments.

REST API: Endpoints for integrating with external requests to add clinics, doctors, and patients and view clinic details.

Tech Stack and Architecture

Tech Stack

- Frontend: HTML, CSS, JavaScript, Bootstrap

- Backend: Django, Django REST framework

- Database: PostgreSQL

Architecture

Models:

Define the structure of the database tables for Clinics, Doctors, Patients, Visits, and Appointments.

Views:

Handles the logic for rendering templates and processing user input. Include class-based views (e.g., ListView, DetailView) and function-based views for specific tasks.

Templates:

HTML files that define the structure of the web pages. Use Django template language to display data dynamically.

URLs:

Route incoming requests to the appropriate view functions or classes. Separate URL configurations for app-specific routes and API endpoints.

API:

Provides endpoints for CRUD operations on Clinics, Doctors, Patients, Visits, and Appointments. Uses serializers to convert complex data types into JSON.

Admin Interface:

Utilizes Django's built-in admin panel to manage data directly.

Static Files:

This includes CSS and JavaScript files that Django serves during development.

Setup Instructions

1. Clone the Repository

```
git clone https://github.com/NaitikR/dental_management_system.git
cd dental_management_system
```

2. Set Up Virtual Environment:

```
python -m venv venv
# On MacOS use 'source venv/bin/activate'
# On Windows use 'venv\Scripts\activate'
```

3. Install Dependencies:

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

4. Configure PostgreSQL:

Enter Postgres and create a database and a user for the database and grant user access.

```
psql postgres
CREATE DATABASE dental_management_db;
CREATE USER 'username' WITH PASSWORD 'your_password';
GRANT ALL PRIVILEGES ON DATABASE dental_management_db TO 'username';
```

Exit postgres by:

```
\q
```

5. Update dental/settings.py with above database and user details:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'dental_management_db',
        'USER': 'your_username',
        'PASSWORD': 'your_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

6. Run Migrations:

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

7. Create Superuser:

```
python manage.py createsuperuser
#You will be prompted to enter a username, email address, and password. For example:
#Username: adminuser
#Email: adminuser@dentalproject.com
#Password: [Enter a secure password]
```

8. Run the Server:

```
python manage.py runserver
```

9. Access the Application:

Visit http://127.0.0.1:8000/admin/ to log in as admin and add sample data.

10. Adding Sample Data

Log in to Admin Interface: Use the superuser credentials.
Add Data: Add clinics, doctors, and patients using the admin interface.

11. Check individual pages of clinics, patients and doctors by using the 'View Site' option on the top right.

You can see lists of the added data, view them, edit them, and add new data from here too.

12. For the REST APIs usage:

I used Postman to check for different API requests, examples of which are given below.

REST API Endpoints

1. Add Patient

Request Method: POST
URL: http://127.0.0.1:8000/api/patients/

Body (JSON):

```
{
  "name": "John Doe",
  "date_of_birth": "1990-01-01",
  "address": "123 Main St",
  "phone_number": "1234567890",
  "ssn_last_4": "1234",
  "gender": "M"
}
```

2. Add Doctor

Method: POST
URL: http://127.0.0.1:8000/api/doctors/

Body (JSON):

```
{
  "npi": "1234567890",
  "name": "Dr. Smith",
  "email": "dr.smith@example.com",
  "phone_number": "0987654321",
  "specialties": ["Cleaning", "Filling"]
}
```

3. Add Clinic

Method: POST
URL: http://127.0.0.1:8000/api/clinics/

Body (JSON):

```
{
  "name": "Downtown Clinic",
```

```
  "phone_number": "1231231234",
  "email": "contact@downtownclinic.com",
  "address": "456 Elm St",
  "city": "Metropolis",
  "state": "NY"
}
```

4. Get Clinic Information

```
Method: GET
URL: http://127.0.0.1:8000/api/clinics/
#this will give the list of all the clinics with their ids, then you can request individual clinics using id:
URL: http://127.0.0.1:8000/api/clinics/<id>/
```

Assumptions Made

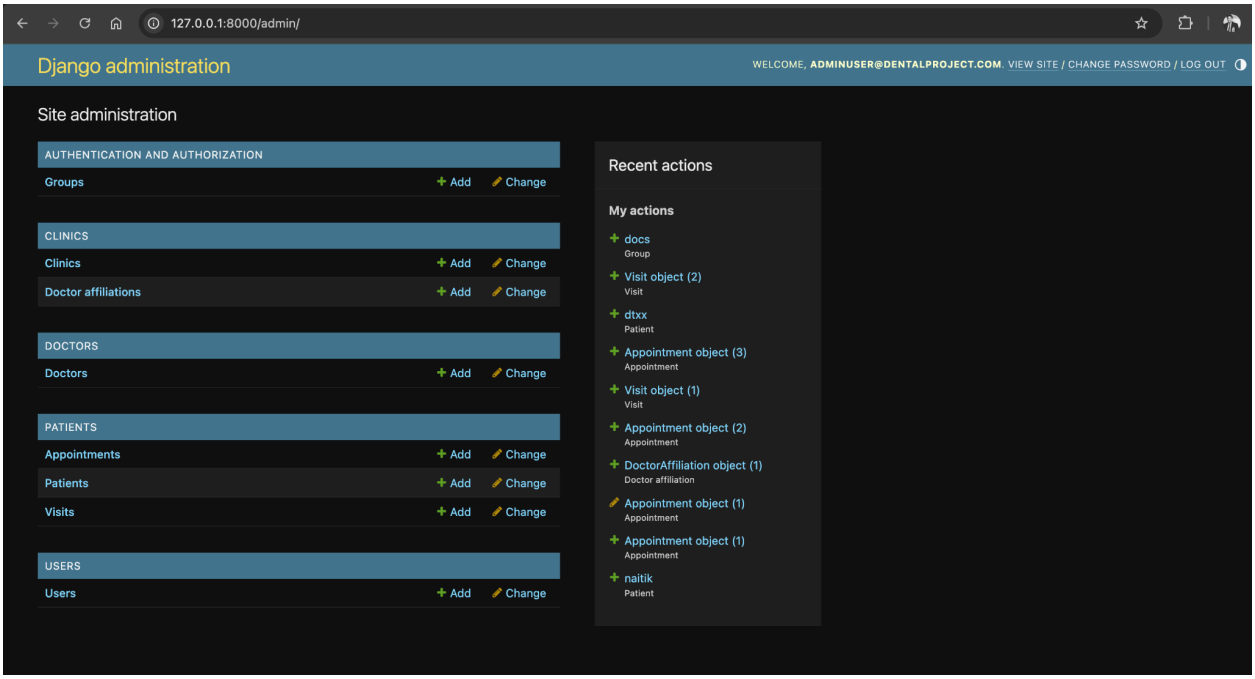
- The user interface is intuitive for administrative users.
- Clinics must have at least one doctor to offer procedures.
- Procedures are predefined and linked to doctor specialties.

Testing

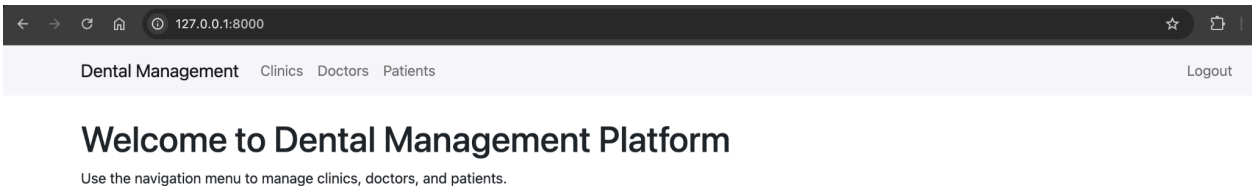
- Unit tests for critical functionalities can be added like:
- User authentication when multiple users are registered.
- CRUD operations for different clinics, patients, and doctors.
- Appointment scheduling for patients with different doctors, treatments and clinics.

Screenshots for the Application

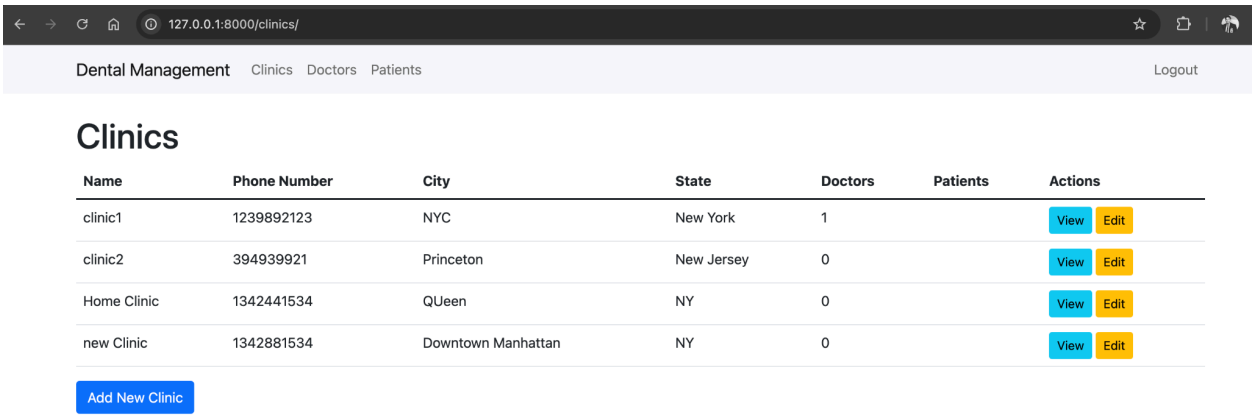
Admin Page



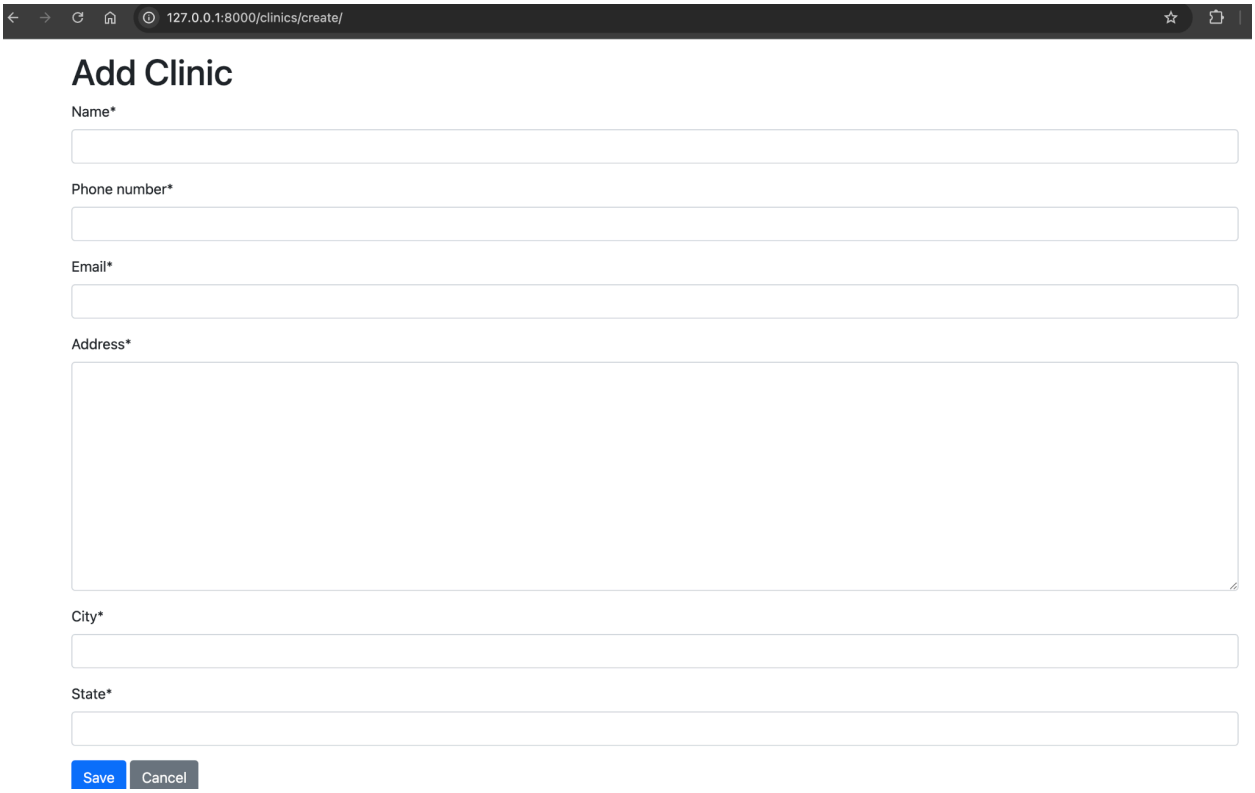
Home Page



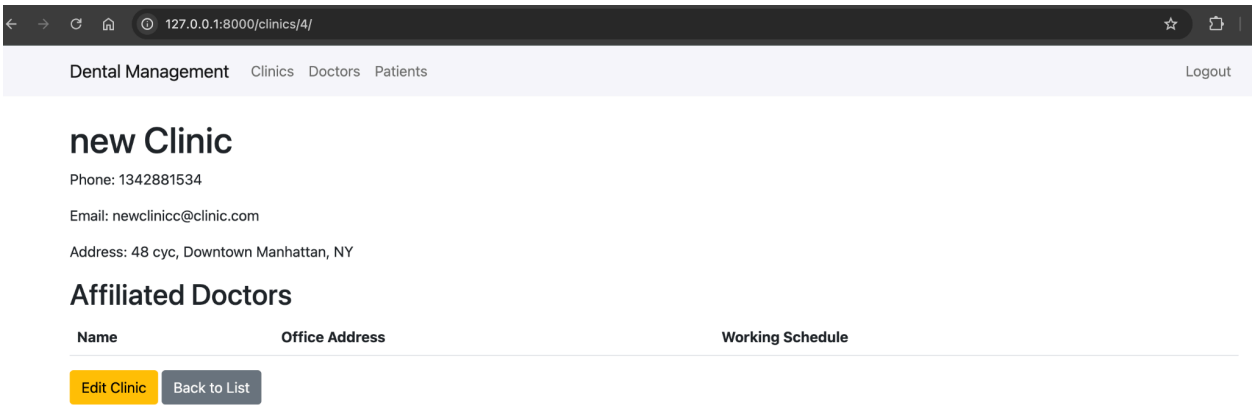
Clinics Page



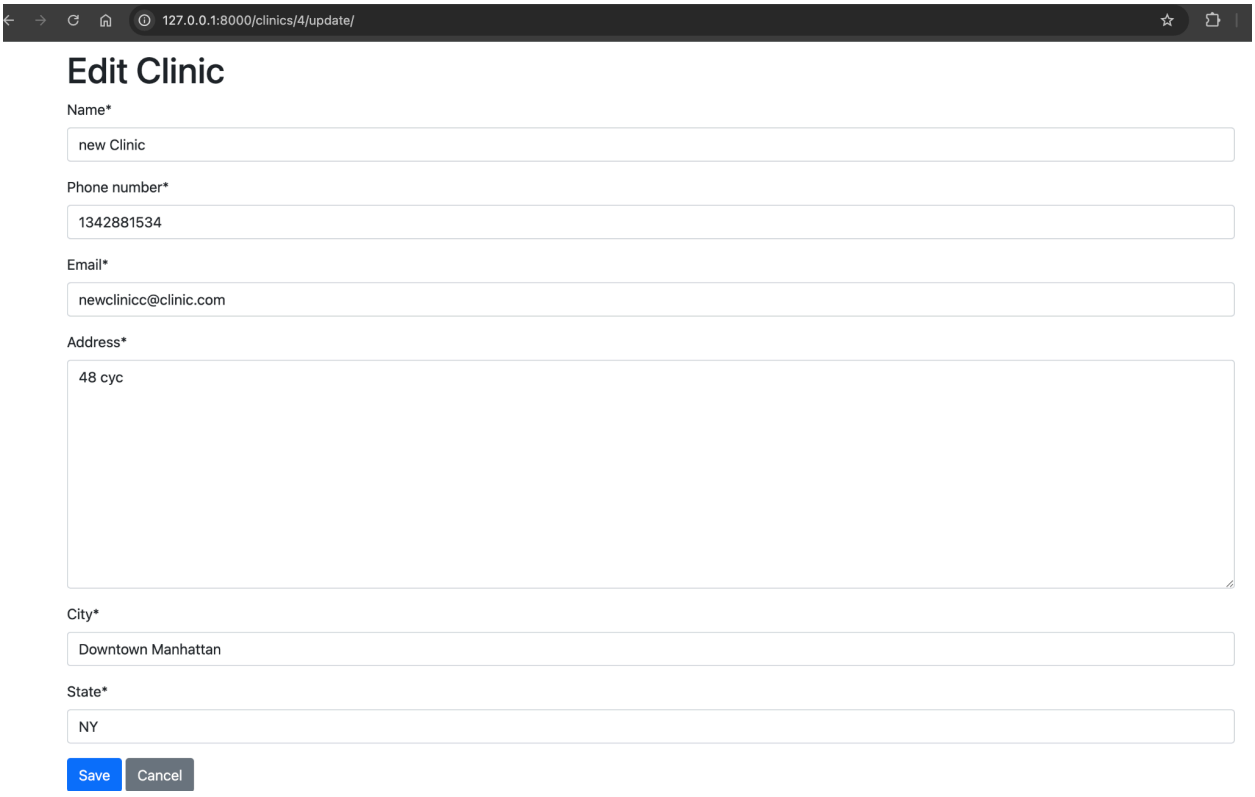
Add Clinic



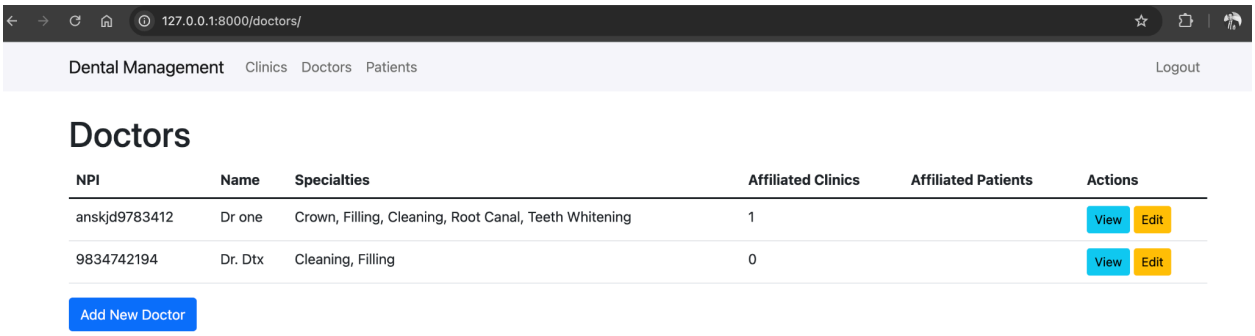
Clinic details



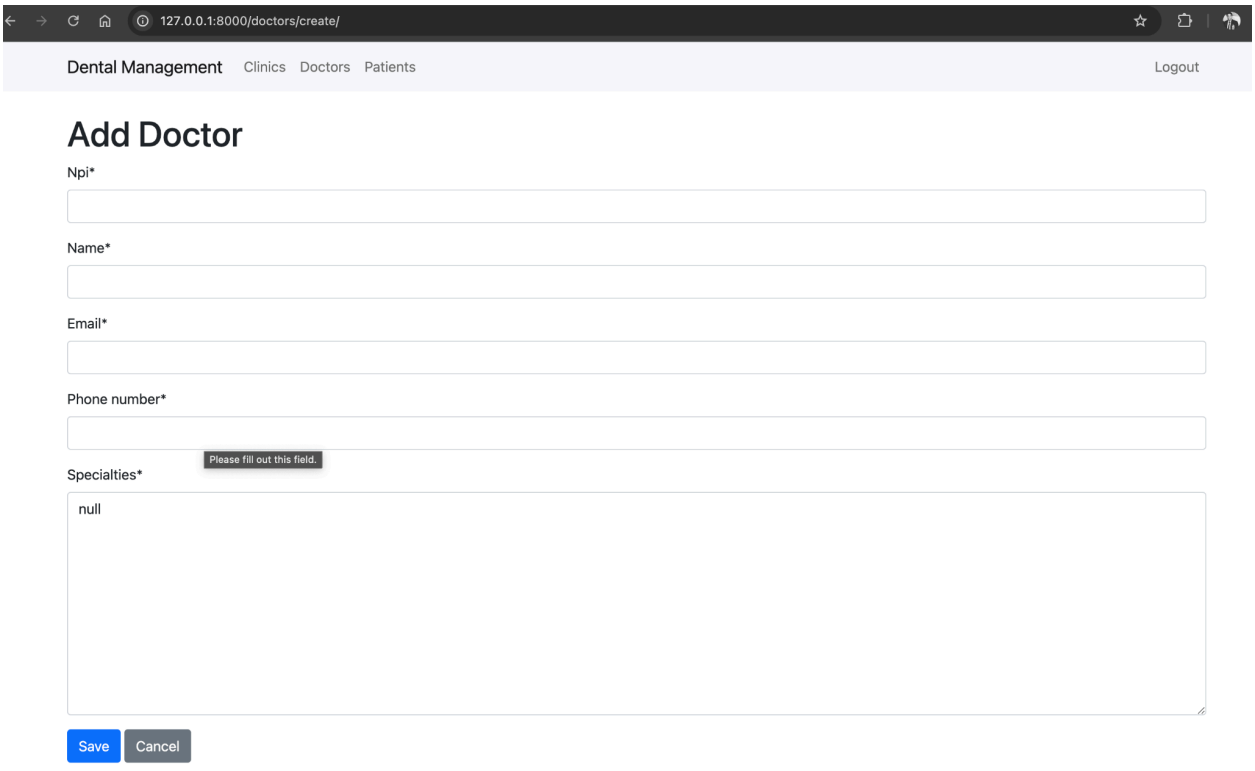
Edit Clinic Details



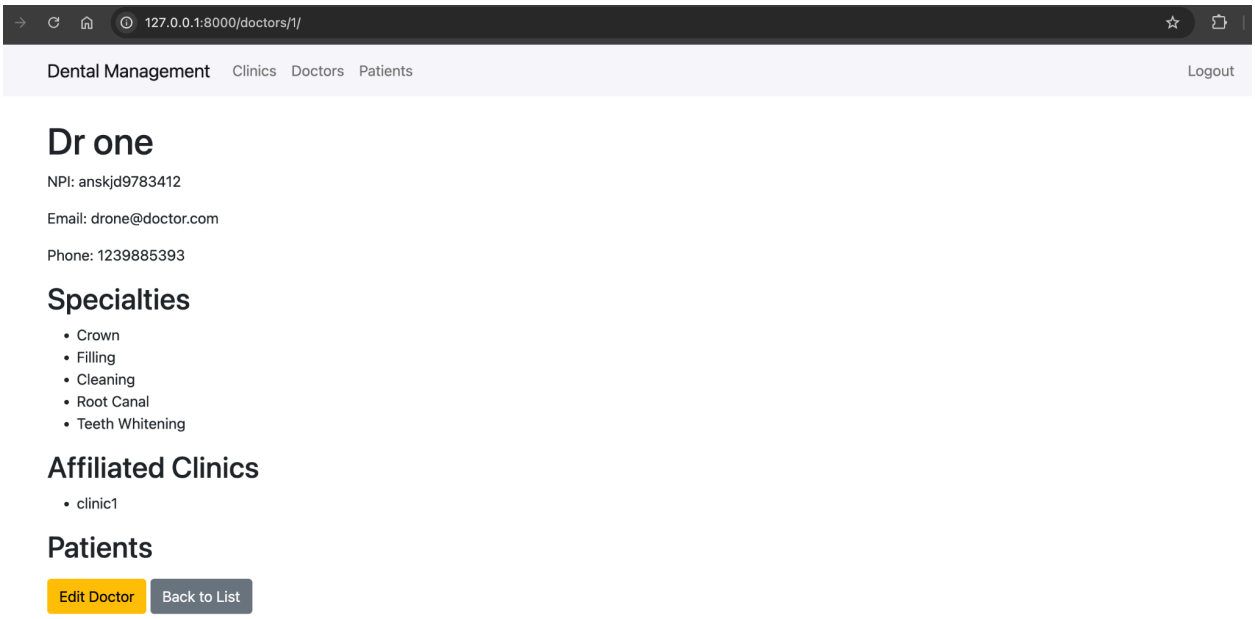
Doctors list page



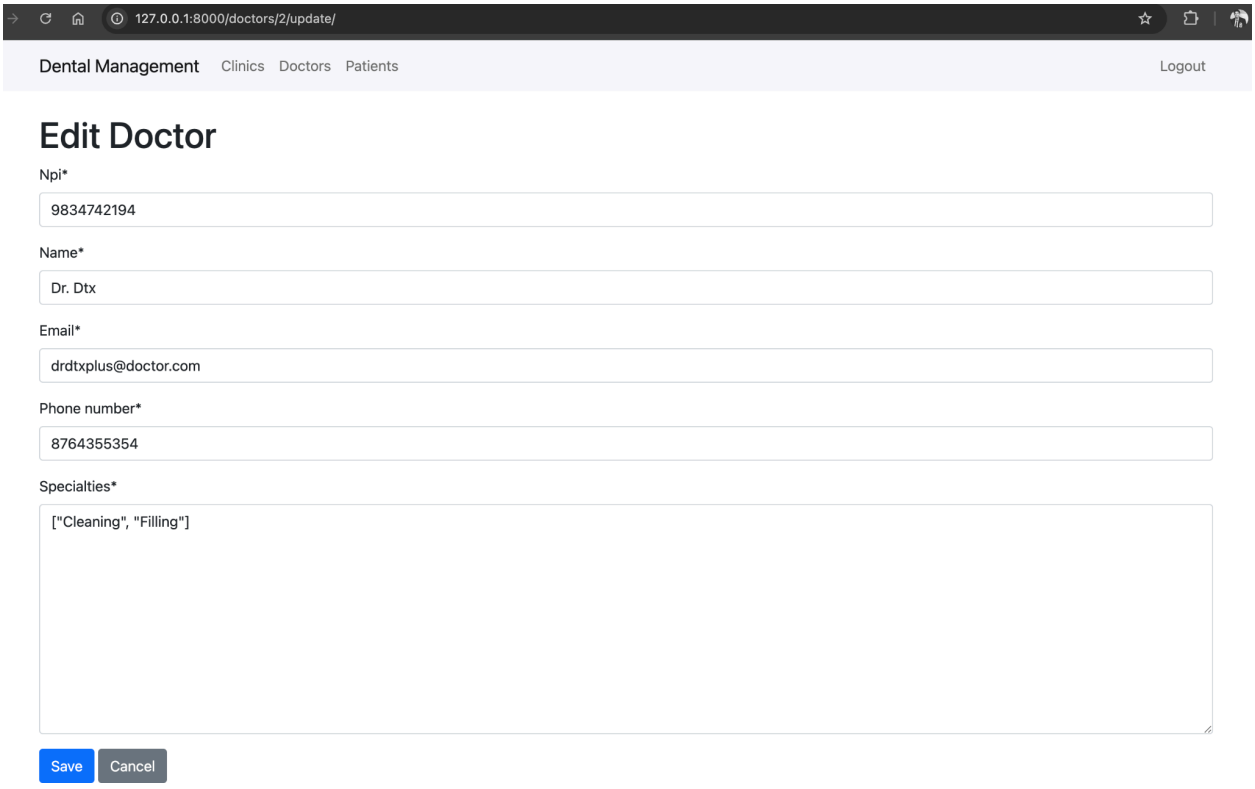
Add New Doctor



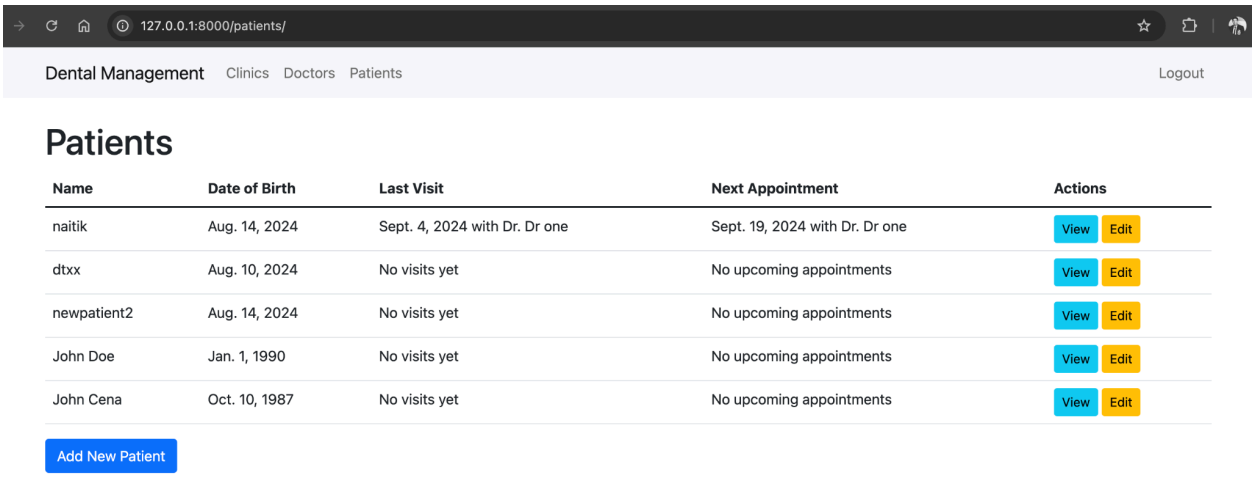
Doctor details page



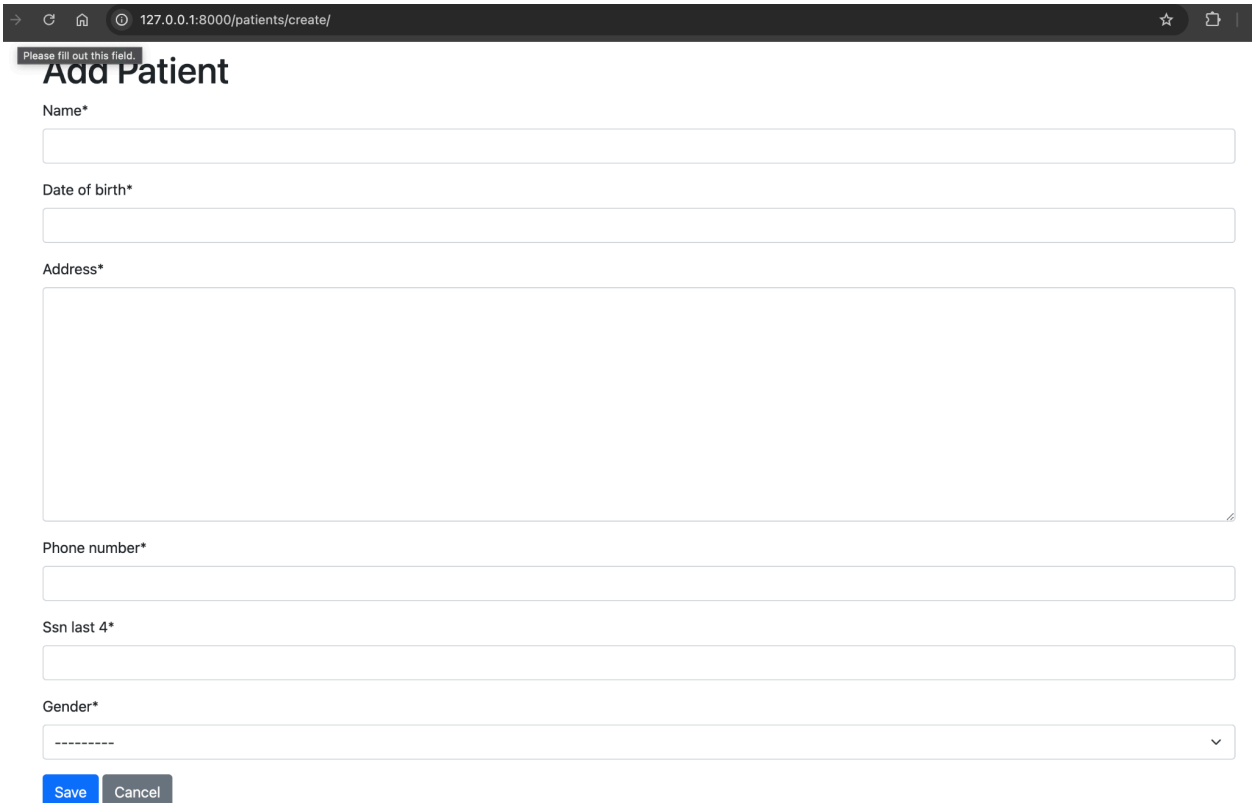
Edit doctor details:



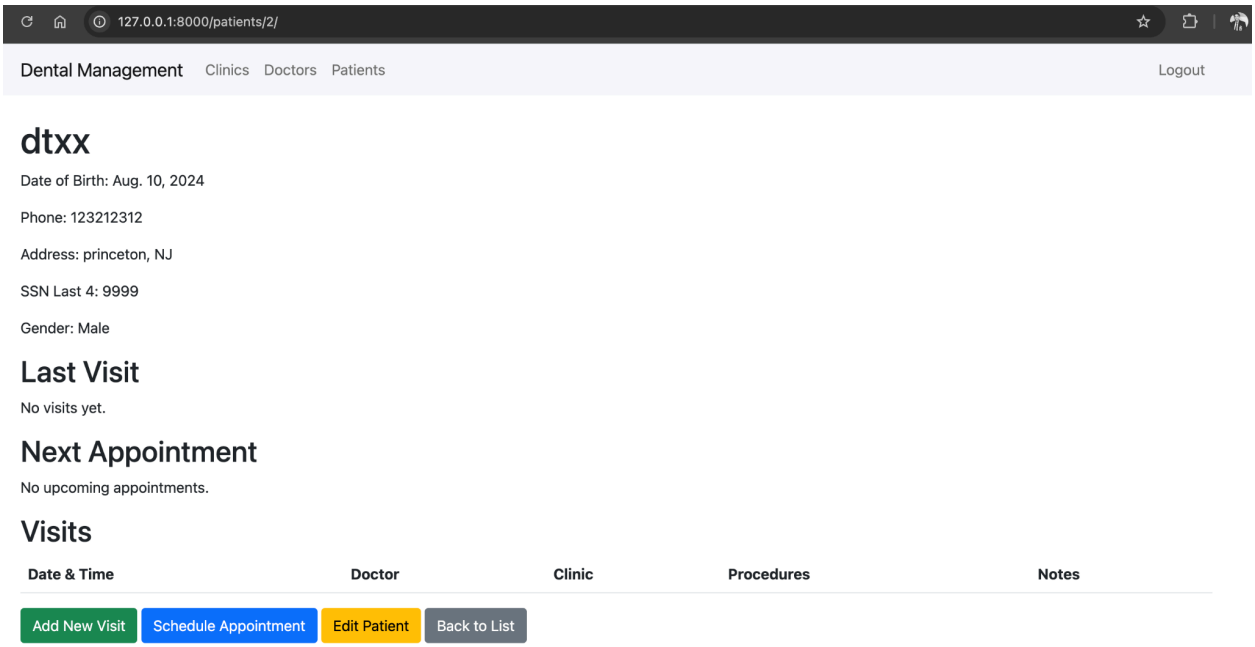
Patients List Page



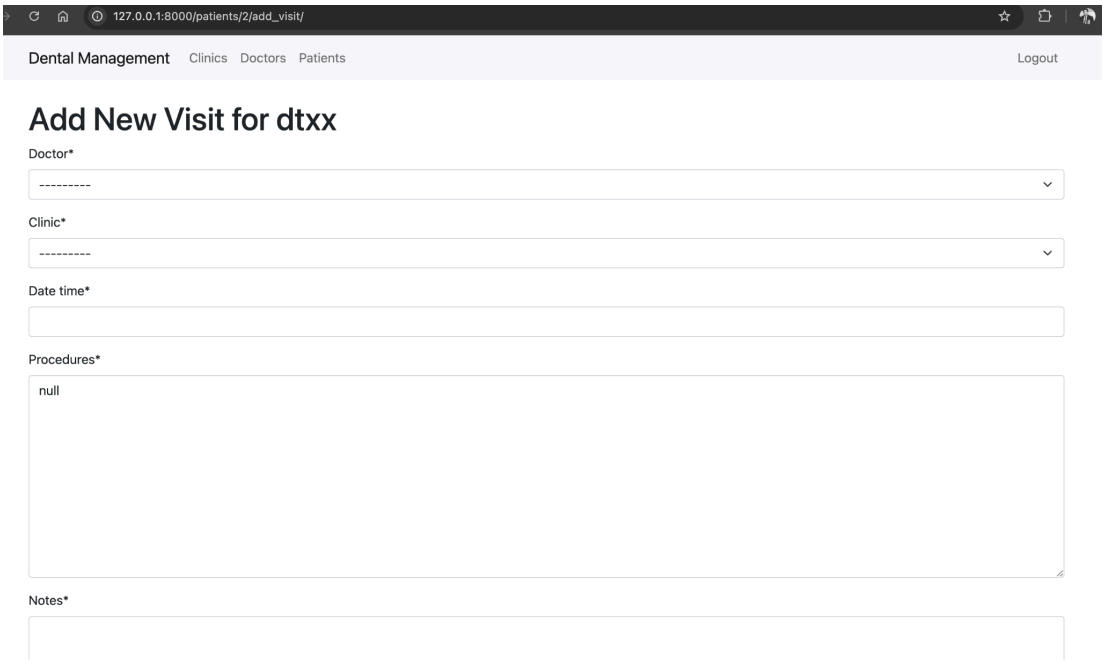
Add new Patient



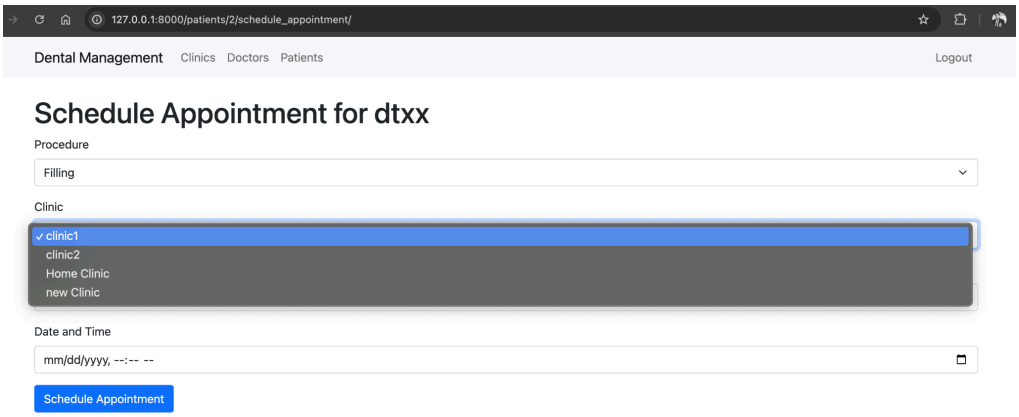
Patient details page

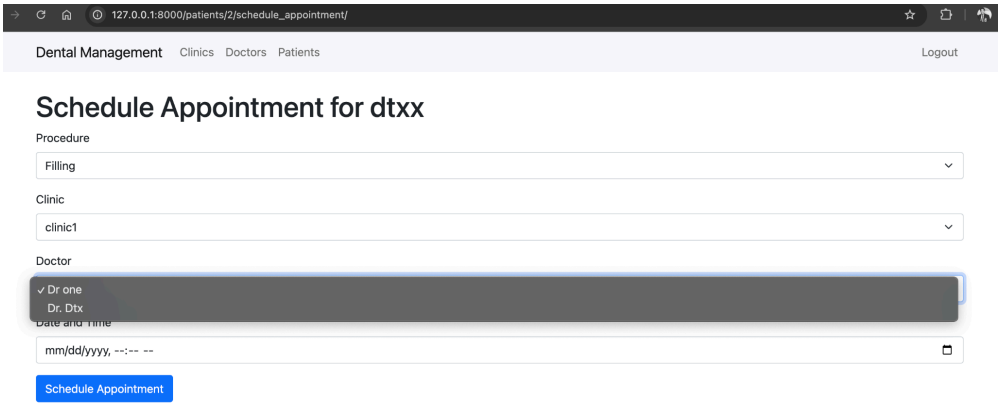


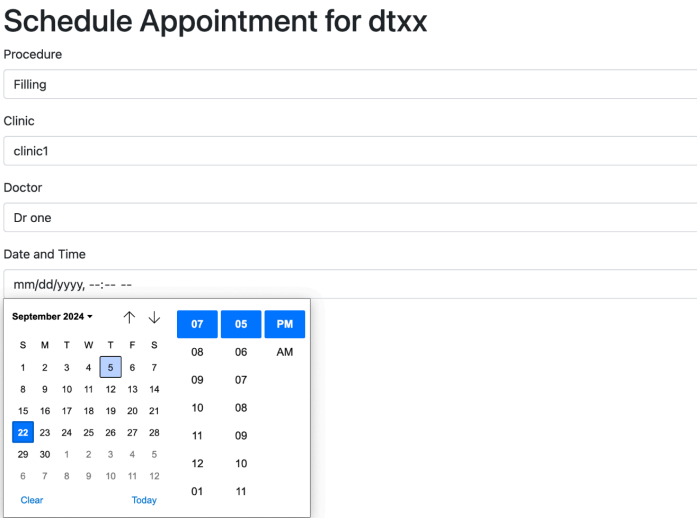
Add new visit for the patient



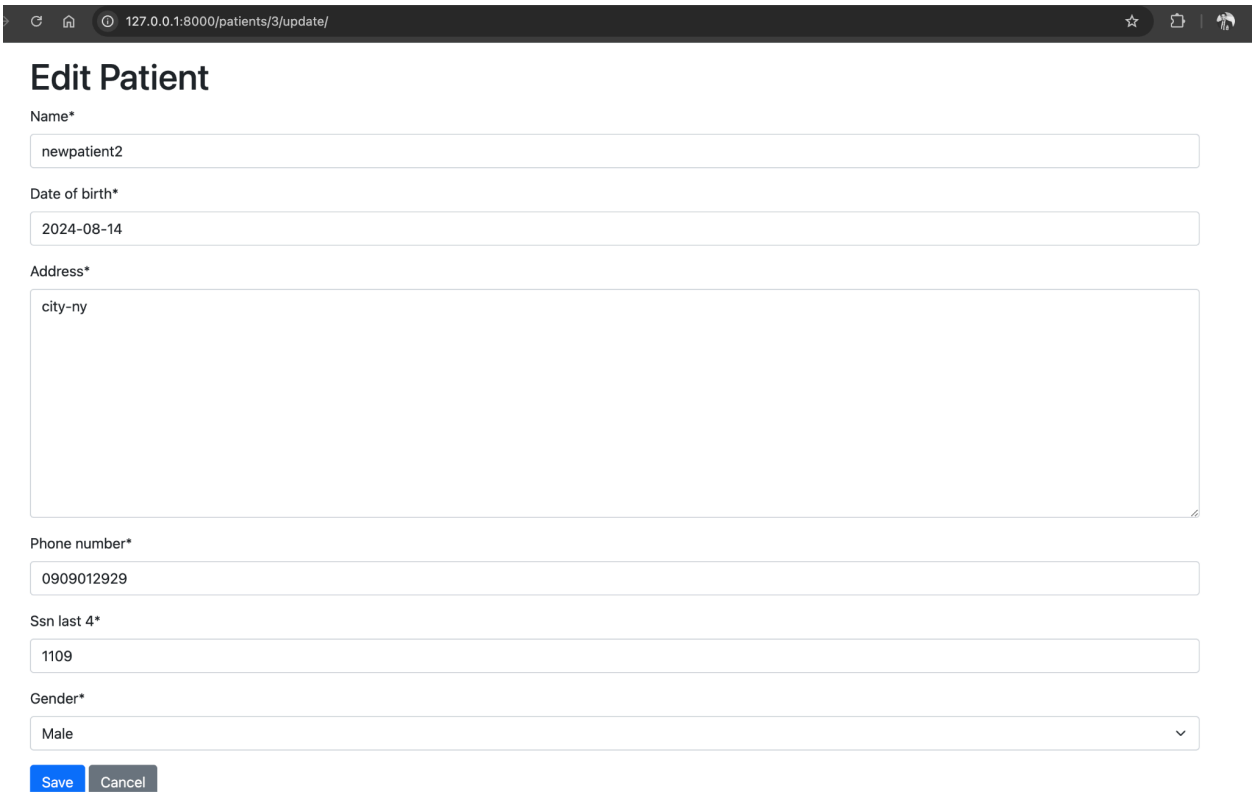
Schedule a new appointment for patient





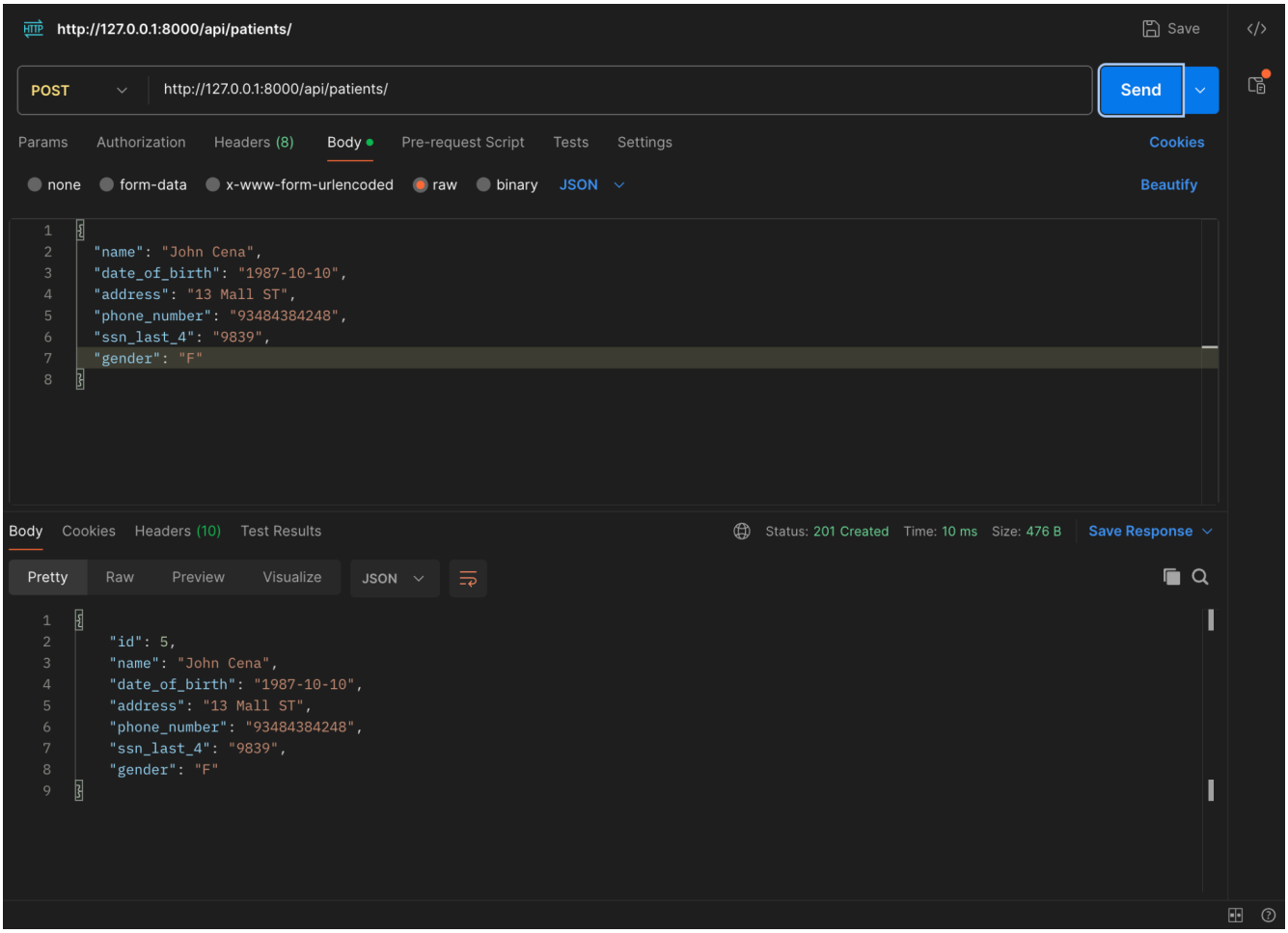


Edit patient details

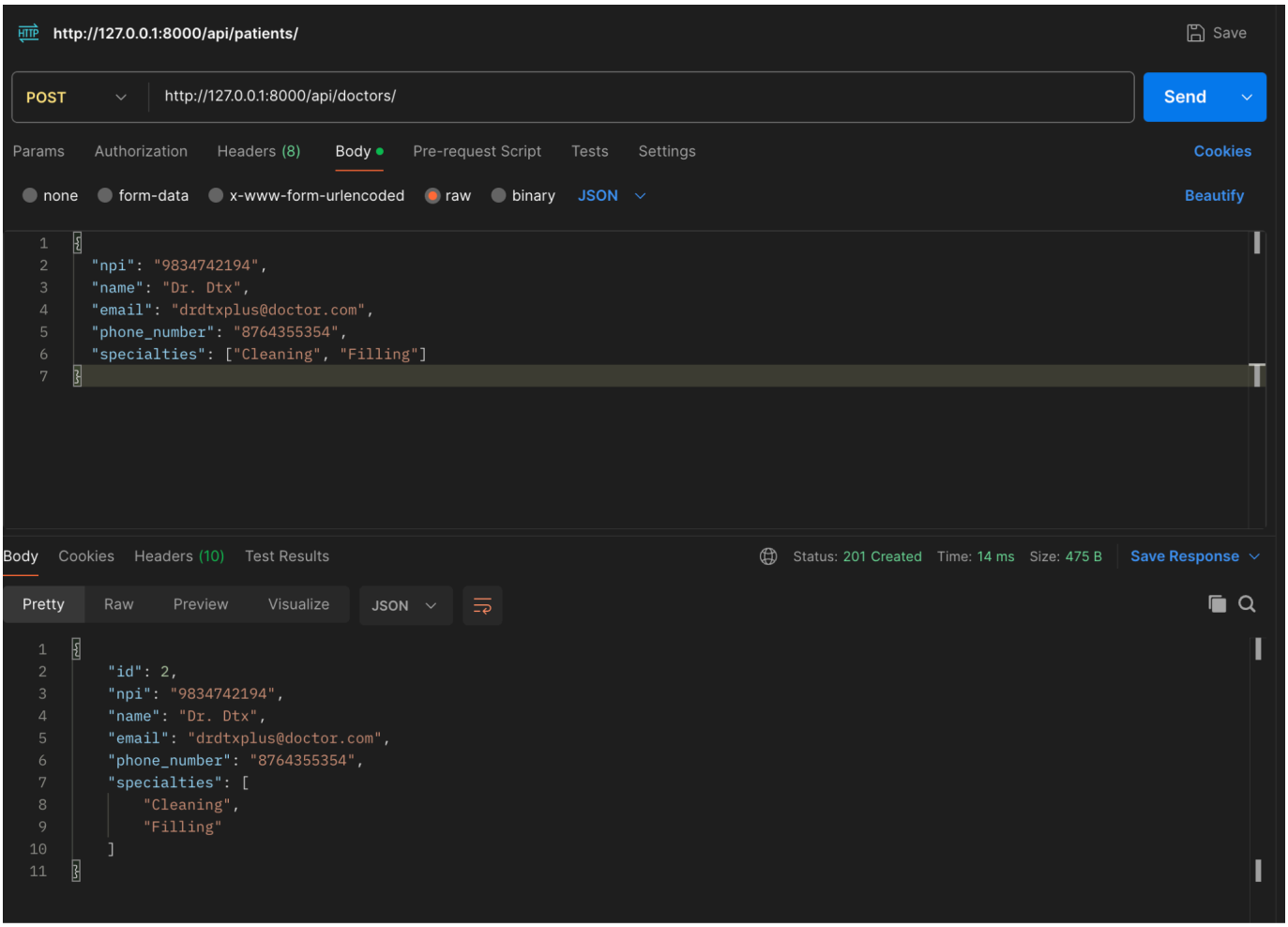


REST API Endpoints:

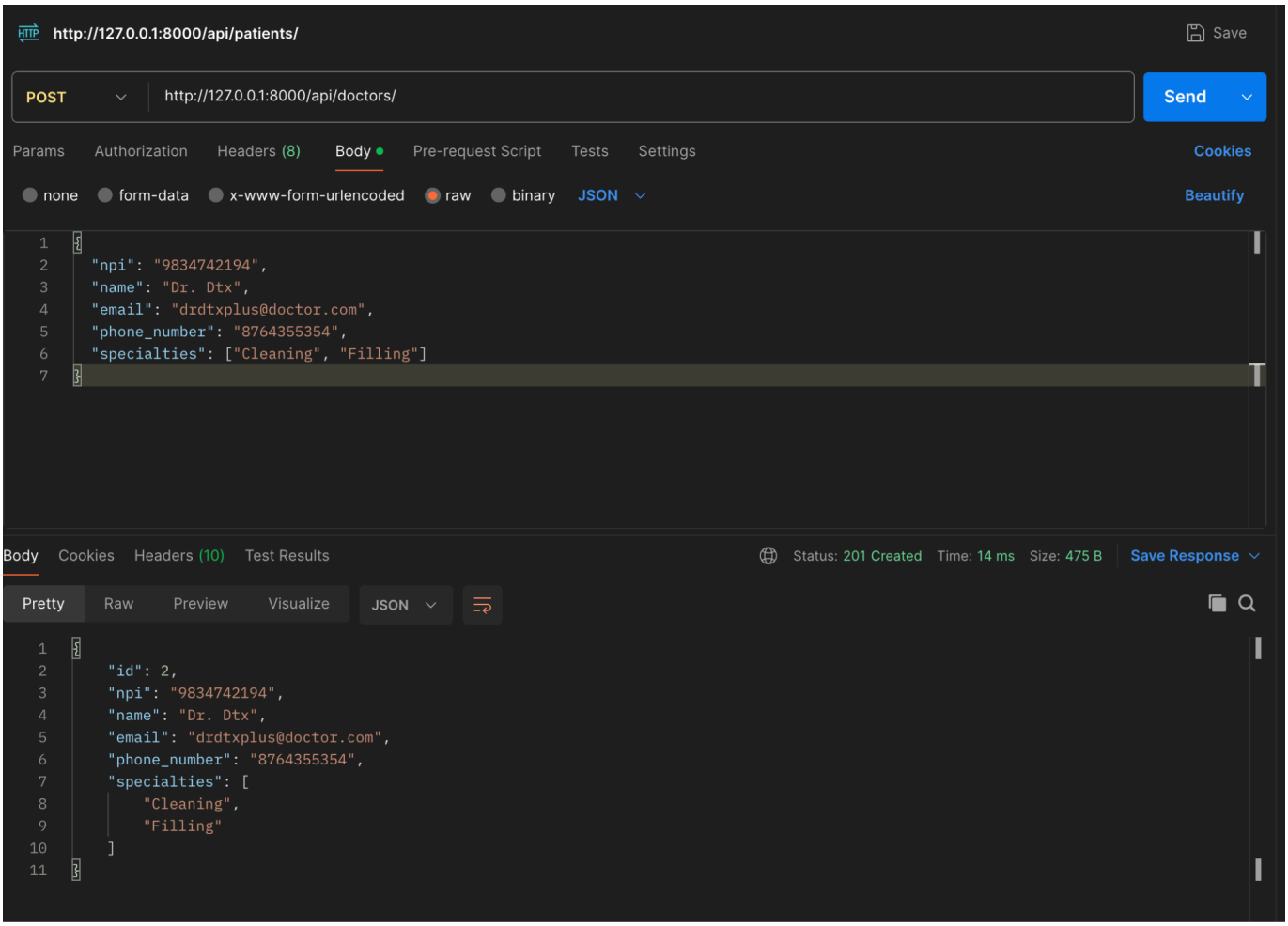
Patient POST request:



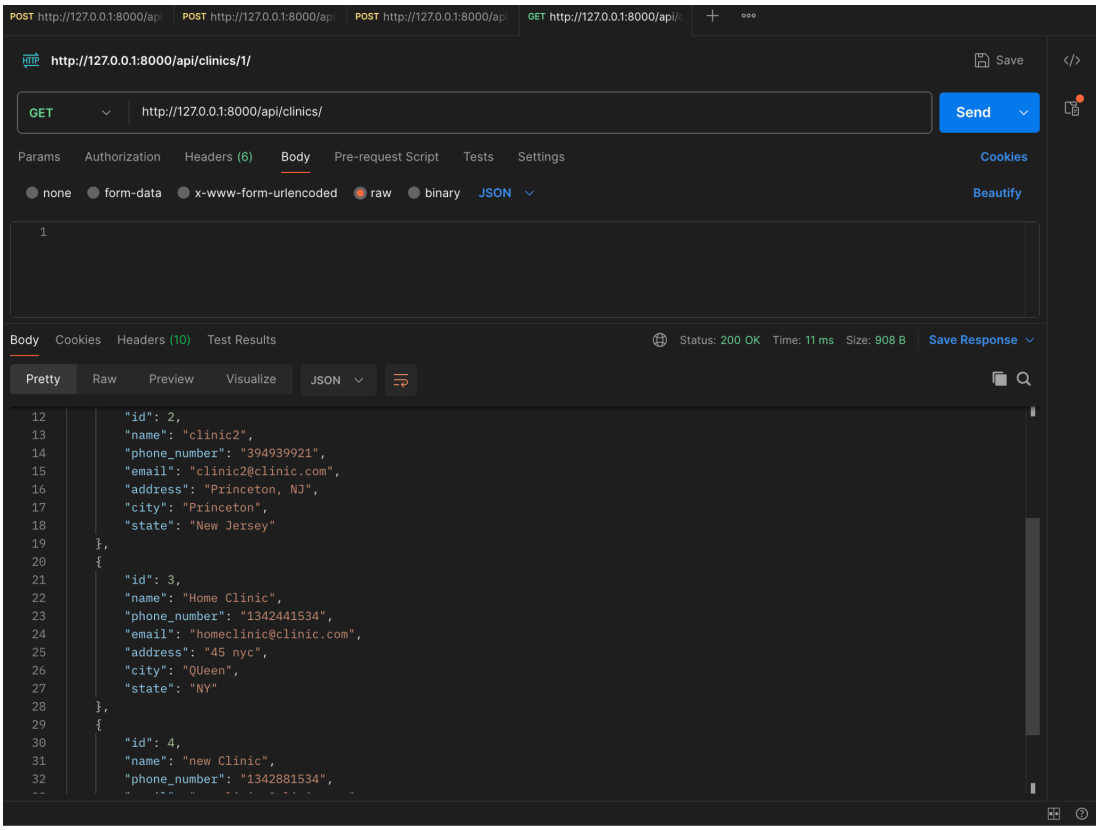
Doctor POST request:

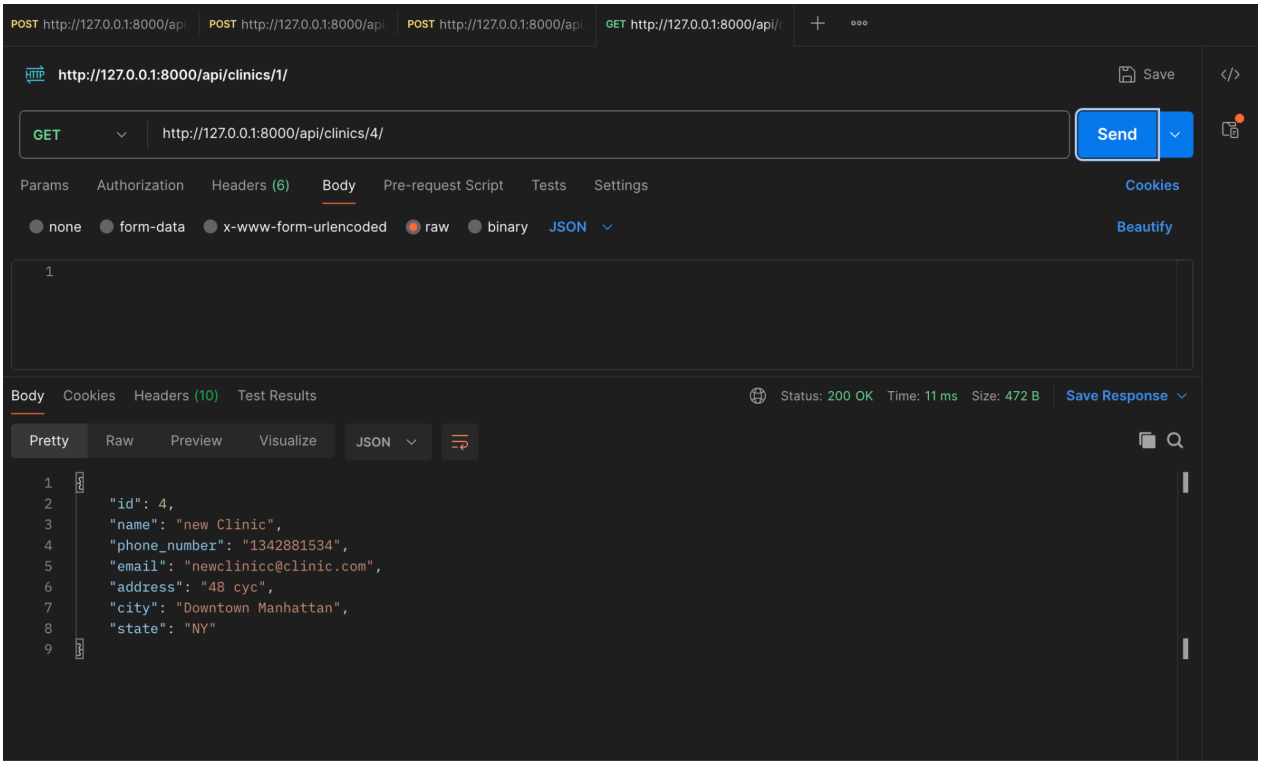


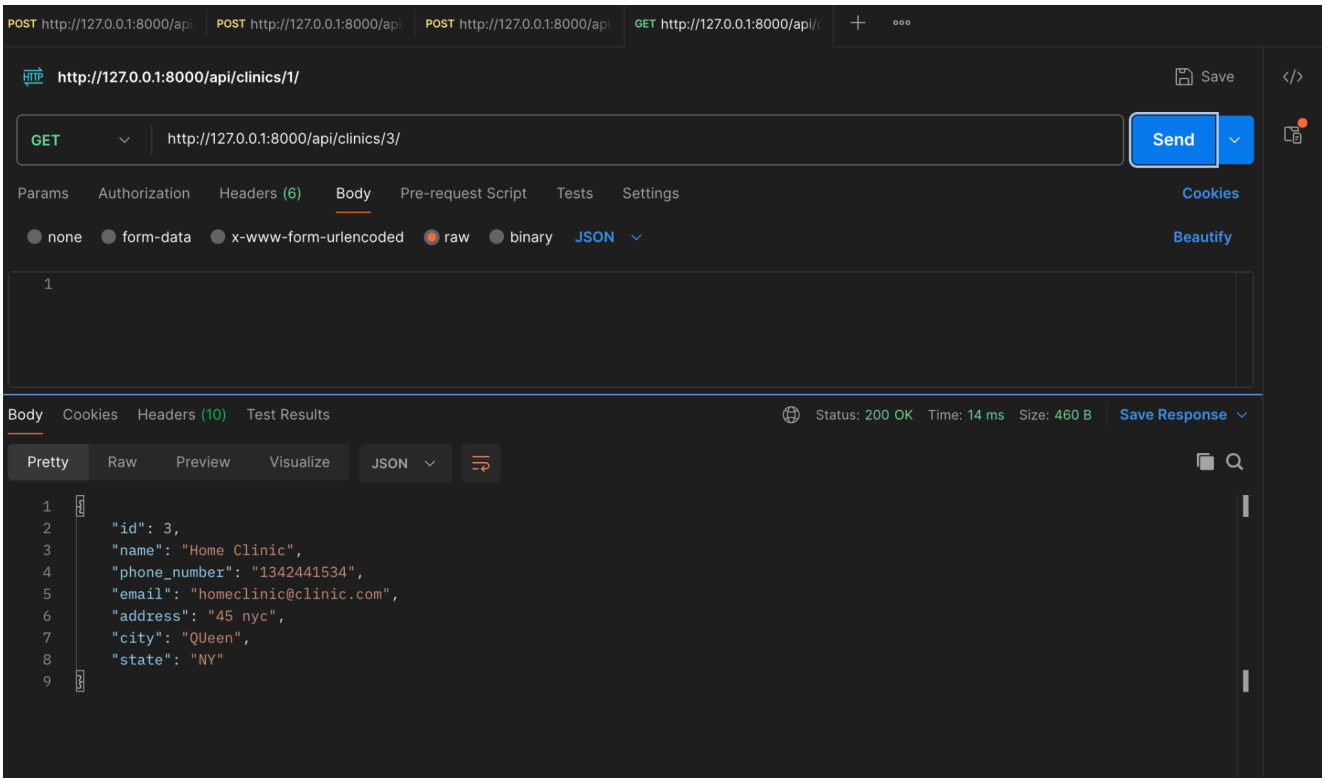
Clinic POST Request



Clinic GET request







Server log for the session:

```
[22/Sep/2024 17:50:17] "POST /api/patients/ HTTP/1.1" 201 140
[22/Sep/2024 17:50:22] "GET /patients/ HTTP/1.1" 200 4523
[22/Sep/2024 17:50:22] "GET /static/css/style.css HTTP/1.1" 404 1853
[22/Sep/2024 17:50:22] "GET /static/js/script.js HTTP/1.1" 404 1850
[22/Sep/2024 18:02:29] "POST /api/clinics/ HTTP/1.1" 201 136
[22/Sep/2024 18:03:16] "POST /api/doctors/ HTTP/1.1" 201 140
[22/Sep/2024 22:35:18] "GET / HTTP/1.1" 200 1830
[22/Sep/2024 22:35:18] "GET /static/css/style.css HTTP/1.1" 404 1853
[22/Sep/2024 22:35:18] "GET /static/js/script.js HTTP/1.1" 404 1850
[22/Sep/2024 22:35:18] "GET /admin/ HTTP/1.1" 200 14177
[22/Sep/2024 22:47:28] "POST /api/clinics/ HTTP/1.1" 201 148
[22/Sep/2024 22:48:22] "POST /api/patients/ HTTP/1.1" 201 141
[22/Sep/2024 22:48:41] "GET /api/clinics/ HTTP/1.1" 200 578
[22/Sep/2024 22:49:13] "GET /api/clinics/3/ HTTP/1.1" 200 136
[22/Sep/2024 22:49:24] "GET /api/clinics/4/ HTTP/1.1" 200 148
```

For any questions, feel free to reach me at:

https://www.linkedin.com/in/naitikr/