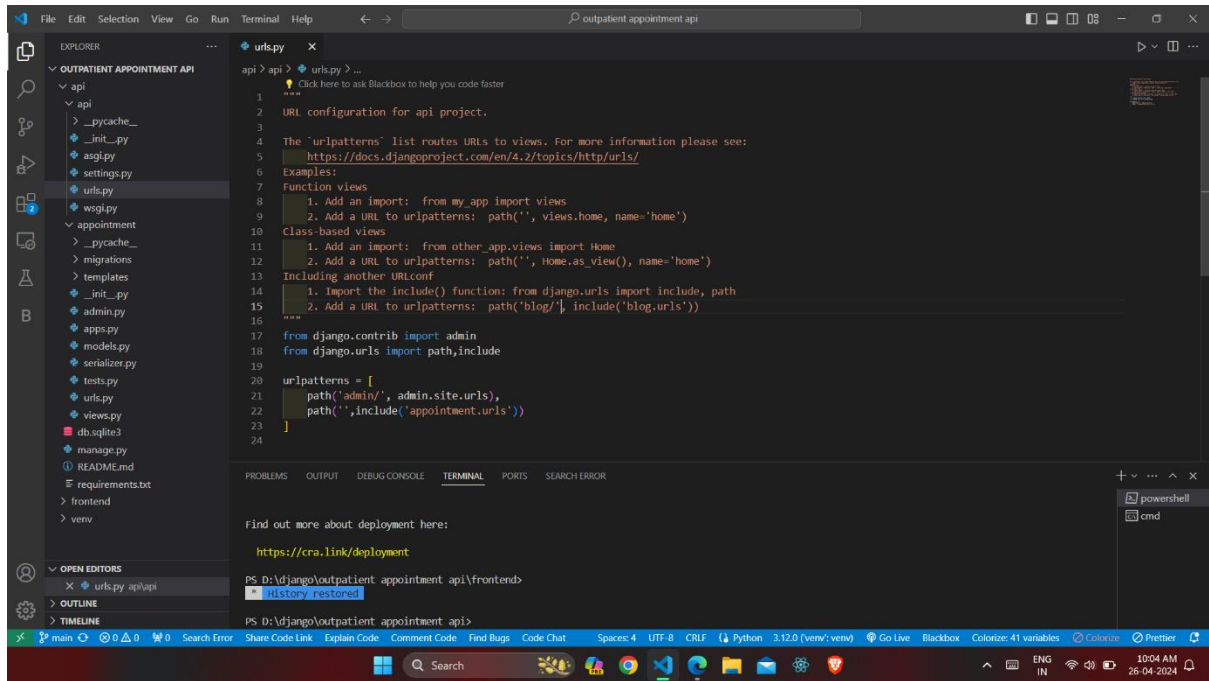


OUTPATIENT APPOINTMENT API

Project Title and Purpose:

- The project is titled “**OUTPATIENT APPOINTMENT API.**”
- It appears to be related to managing outpatient appointments.



```
1  # Django URL Configuration
2  #
3  # The 'urlpatterns' list routes URLs to views. For more information please see:
4  # https://docs.djangoproject.com/en/4.2/topics/http/urls/
5  # Examples:
6  #
7  # Function views
8  # 1. Add an import: from my_app import views
9  # 2. Add a URL to urlpatterns: path('', views.home, name='home')
10 # Class-based views
11 # 1. Add an import: from other_app.views import Home
12 # 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 # Including another URLconf
14 # 1. Import the include() function: from django.urls import include, path
15 # 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('', include('appointment.urls'))
23 ]
```

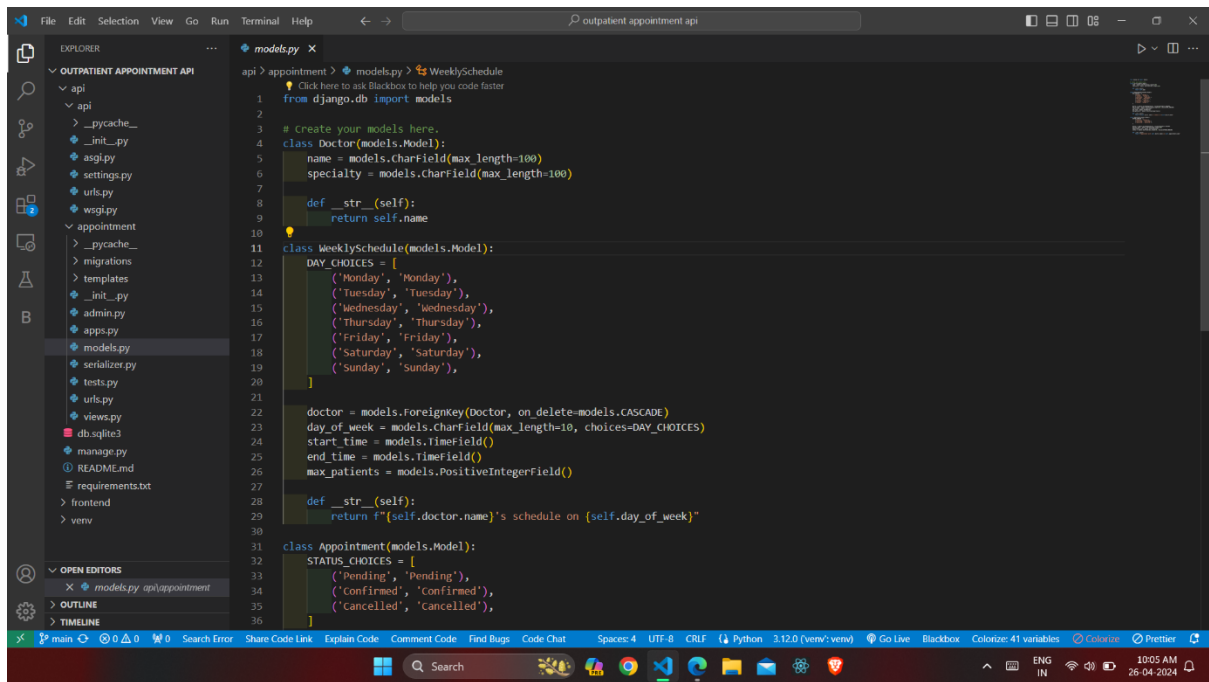
1. URL Configuration (urls.py):

- The file `urls.py` is used for configuring URL patterns in a Django web application.
- The URL patterns map specific URLs to corresponding views (functions or classes).
- The `urlpatterns` list contains routes for different parts of the application.

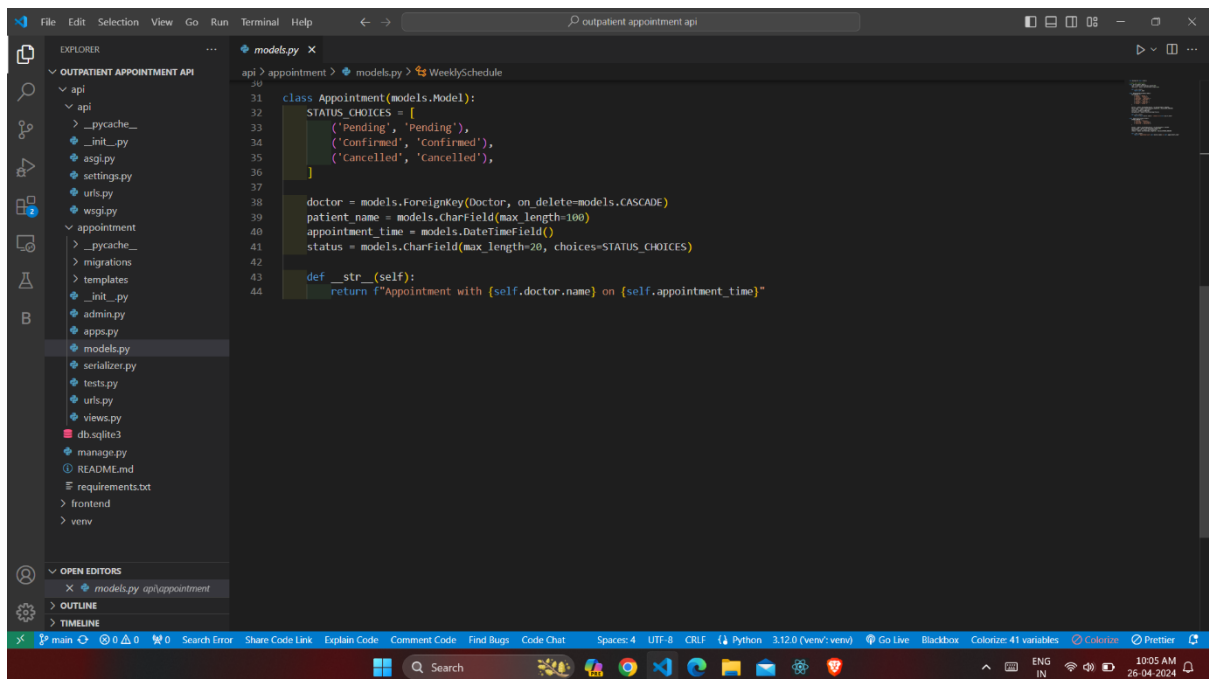
2. Examples of URL Patterns:

- Example 1:
 - URL: `/admin/`
 - Corresponding view: Django admin interface
- Example 2:
 - URL: `/`
 - Corresponding view: Includes the app named “appointment” (defined in `appointment.urls`)

OUTPATIENT APPOINTMENT API



```
1 from django.db import models
2
3 # Create your models here.
4 class Doctor(models.Model):
5     name = models.CharField(max_length=100)
6     specialty = models.CharField(max_length=100)
7
8     def __str__(self):
9         return self.name
10
11 class WeeklySchedule(models.Model):
12     DAY_CHOICES = [
13         ('Monday', 'Monday'),
14         ('Tuesday', 'Tuesday'),
15         ('Wednesday', 'Wednesday'),
16         ('Thursday', 'Thursday'),
17         ('Friday', 'Friday'),
18         ('Saturday', 'Saturday'),
19         ('Sunday', 'Sunday'),
20     ]
21
22     doctor = models.ForeignKey(Doctor, on_delete=models.CASCADE)
23     day_of_week = models.CharField(max_length=10, choices=DAY_CHOICES)
24     start_time = models.TimeField()
25     end_time = models.TimeField()
26     max_patients = models.PositiveIntegerField()
27
28     def __str__(self):
29         return f"{self.doctor.name}'s schedule on {self.day_of_week}"
30
31 class Appointment(models.Model):
32     STATUS_CHOICES = [
33         ('Pending', 'Pending'),
34         ('Confirmed', 'Confirmed'),
35         ('Cancelled', 'Cancelled'),
36     ]
```

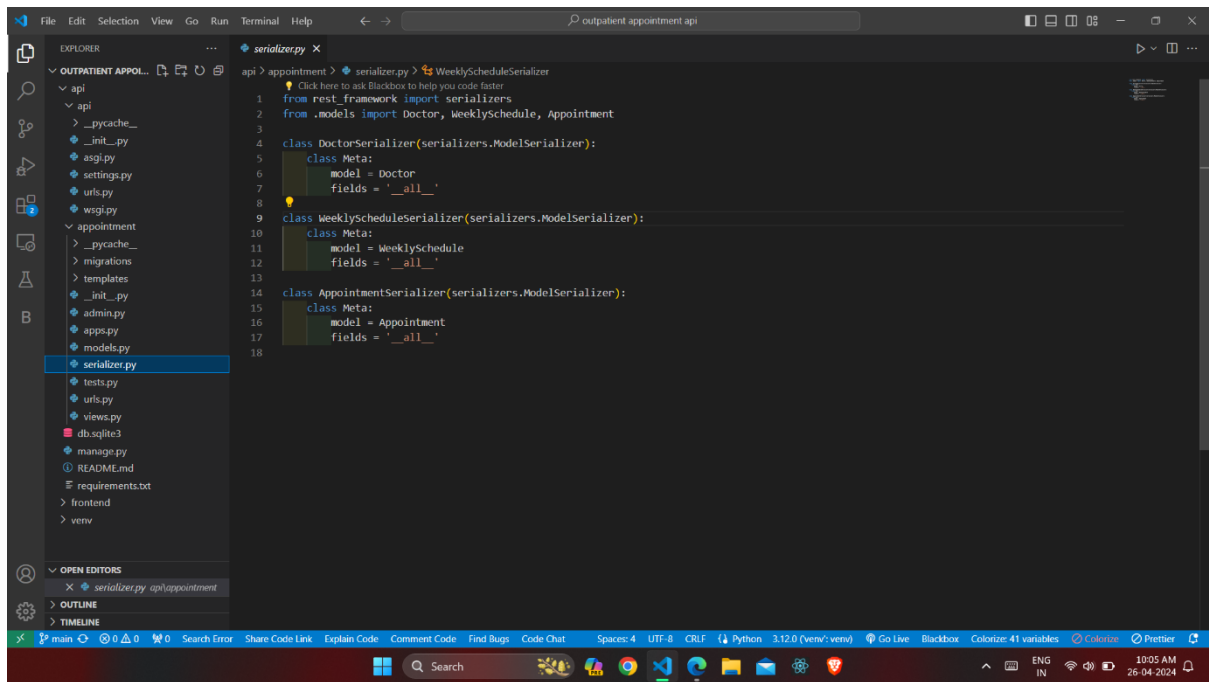


```
37
38     doctor = models.ForeignKey(Doctor, on_delete=models.CASCADE)
39     patient_name = models.CharField(max_length=100)
40     appointment_time = models.DateTimeField()
41     status = models.CharField(max_length=20, choices=STATUS_CHOICES)
42
43     def __str__(self):
44         return f"Appointment with {self.doctor.name} on {self.appointment_time}"
```

1. Models.py:

- This is the Object Relation Mapping (ORM) file.
- This file contains the structure of the database.
- Classes like Doctor, WeeklySchedule and, Appointment are the tables that explaining the structure of the tables.

OUTPATIENT APPOINTMENT API

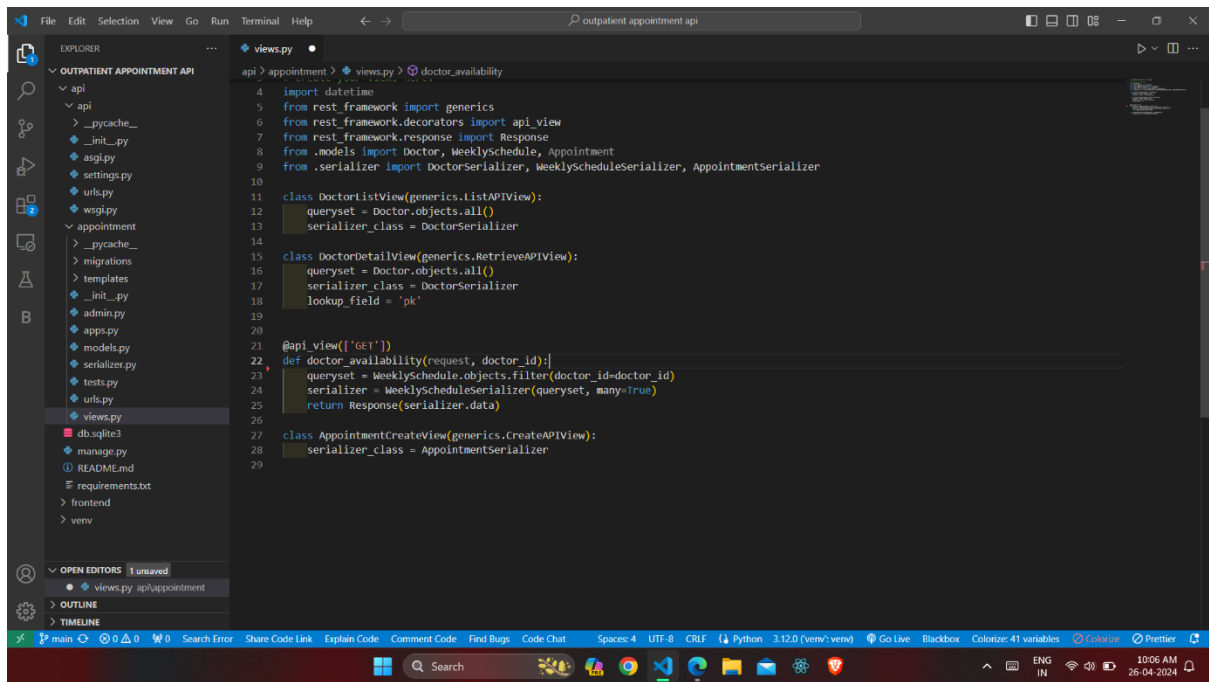


Serializer.py

- DoctorSerializer:**
 - This serializer class is for the `Doctor` model. It includes all fields of the model in the serialization process.
- WeeklyScheduleSerializer:**
 - This serializer class handles the `WeeklySchedule` model, serializing all its fields.
- AppointmentSerializer:**
 - This serializer class is for the `Appointment` model, including all fields in the serialization.

Each `Meta` class within the serializers specifies the model it serializes and the fields to include. The `'__all__'` value indicates that all fields in the respective model should be included in the serialized data. These serializers are typically used to convert complex data types to Python data types that can then be easily rendered into JSON for API responses.

OUTPATIENT APPOINTMENT API



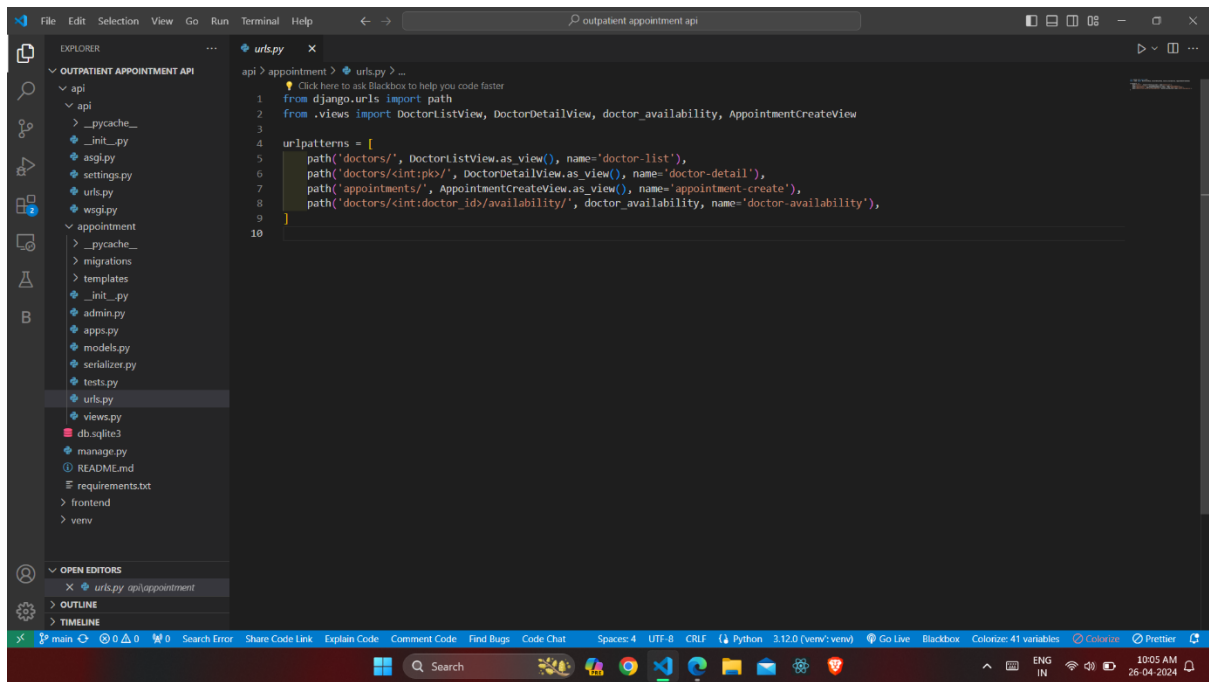
```
api > appointment > views.py > doctor_availability
4 import datetime
5 from rest_framework import generics
6 from rest_framework.decorators import api_view
7 from rest_framework.response import Response
8 from .models import Doctor, WeeklySchedule, Appointment
9 from .serializer import DoctorSerializer, WeeklyScheduleSerializer, AppointmentSerializer
10
11 class DoctorListView(generics.ListAPIView):
12     queryset = Doctor.objects.all()
13     serializer_class = DoctorSerializer
14
15 class DoctorDetailView(generics.RetrieveAPIView):
16     queryset = Doctor.objects.all()
17     serializer_class = DoctorSerializer
18     lookup_field = 'pk'
19
20
21 @api_view(['GET'])
22 def doctor_availability(request, doctor_id):
23     queryset = WeeklySchedule.objects.filter(doctor_id=doctor_id)
24     serializer = WeeklyScheduleSerializer(queryset, many=True)
25     return Response(serializer.data)
26
27 class AppointmentCreateView(generics.CreateAPIView):
28     serializer_class = AppointmentSerializer
29
```

Views.py

1. **DoctorAvailability:** This class inherits from `generics.ListAPIView` and is set up to list all doctor objects using `DoctorSerializer`.
2. **DoctorDetail:** Inherits from `generics.RetrieveAPIView` and retrieves a specific doctor by their primary key (`pk`), again using `DoctorSerializer`.
3. **doctor_availability Function:** Defined with the `@api_view` decorator for HTTP GET requests, it filters the `WeeklySchedule` by `doctor_id`, serializes the data, and returns it in the response.
4. **AppointmentCreateView:** A class for creating new appointments, inheriting from `generics.CreateAPIView` and using `AppointmentSerializer`.

This script is part of the backend API, allowing front-end applications to interact with the database to retrieve doctors' availability and manage appointments. The use of class-based views and function-based view demonstrates Django's flexibility in handling different types of HTTP requests.

OUTPATIENT APPOINTMENT API



Urls.py:

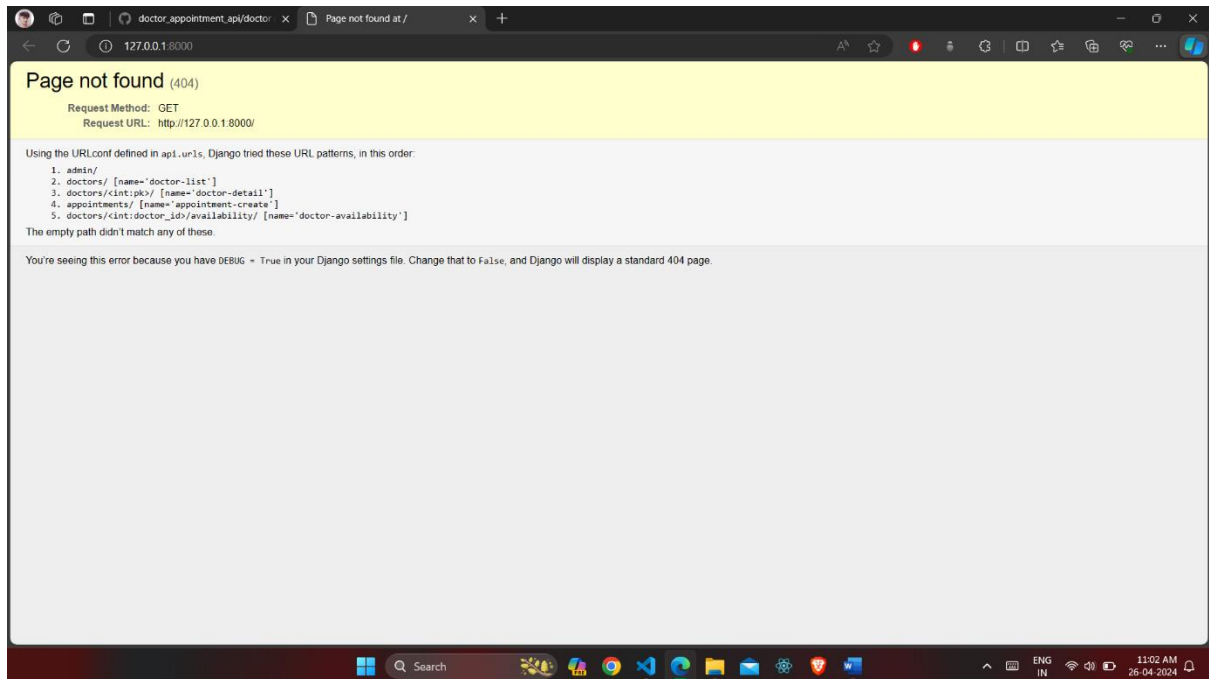
1. URL Patterns:

- A list named `urlpatterns` contains the URL patterns for the app.
- **List View:** The URL ending in `'doctors/'` is mapped to `DoctorListView`, which likely displays a list of doctors.
- **Detail View:** A URL like `'doctors/5/'` (where `'5'` is a doctor's ID) is directed to `DoctorDetailView`, showing details for a specific doctor.
- **Appointment Creation:** Adding `'appointment/'` to the URL directs to `AppointmentCreateView` for scheduling an appointment with that doctor.
- **Availability:** The URL ending in `'availability/'` maps to `doctor_availability`, likely showing when the doctor is available for appointments.

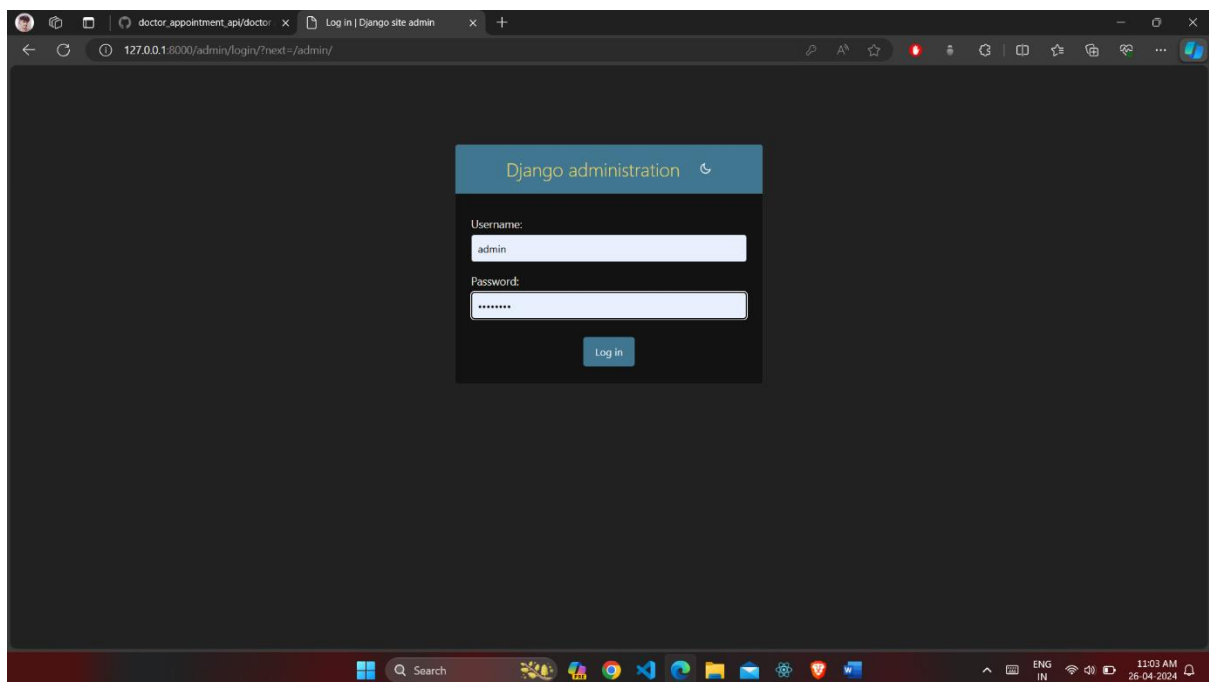
This setup helps users navigate to different parts of the application to view doctors, their details, schedule appointments, and check availability.

OUTPATIENT APPOINTMENT API

Output:

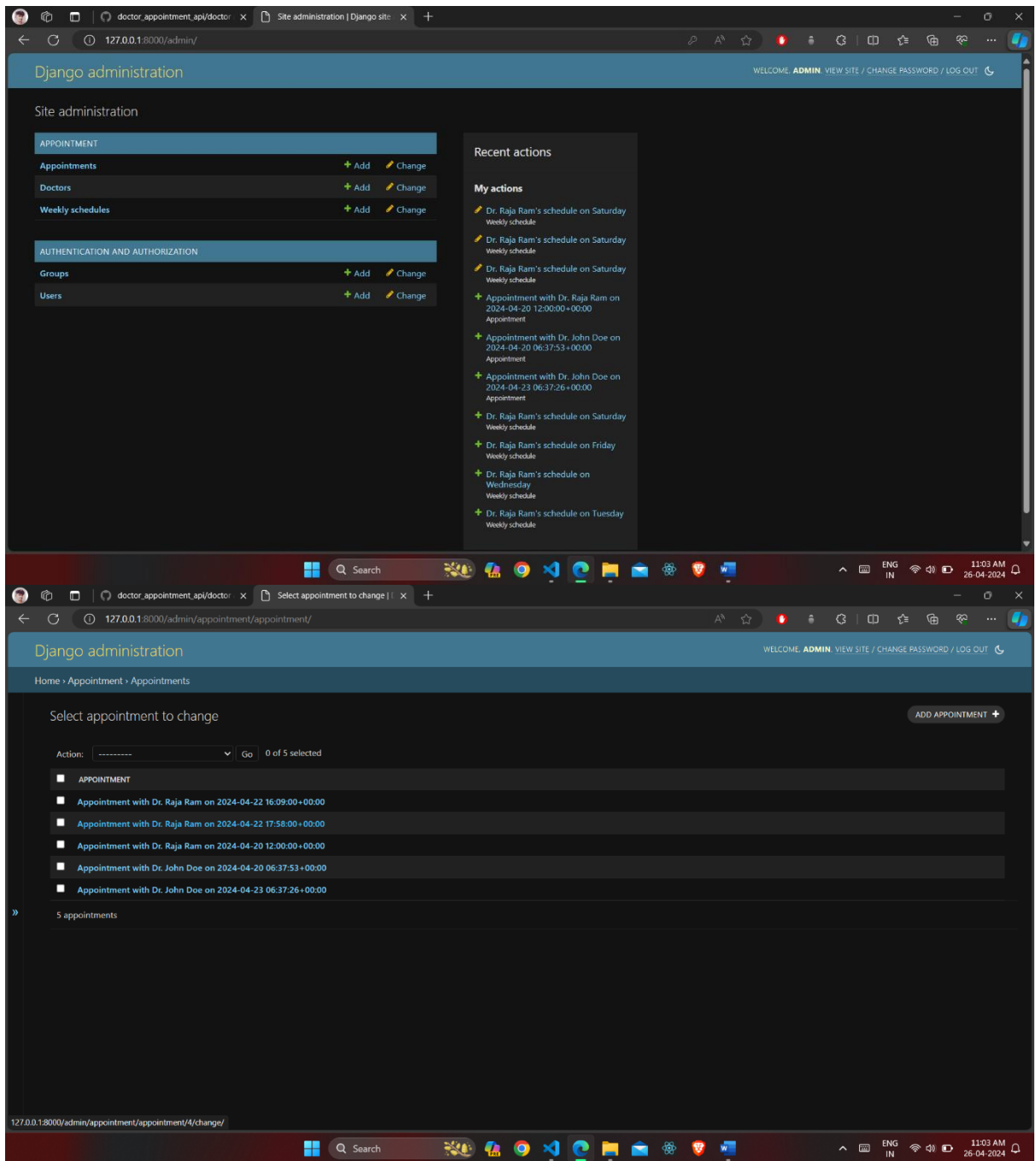


- After running the application this page will appear this is the default page that shows while URL is not found in the project and,
- It shows all the URLs user in this project.

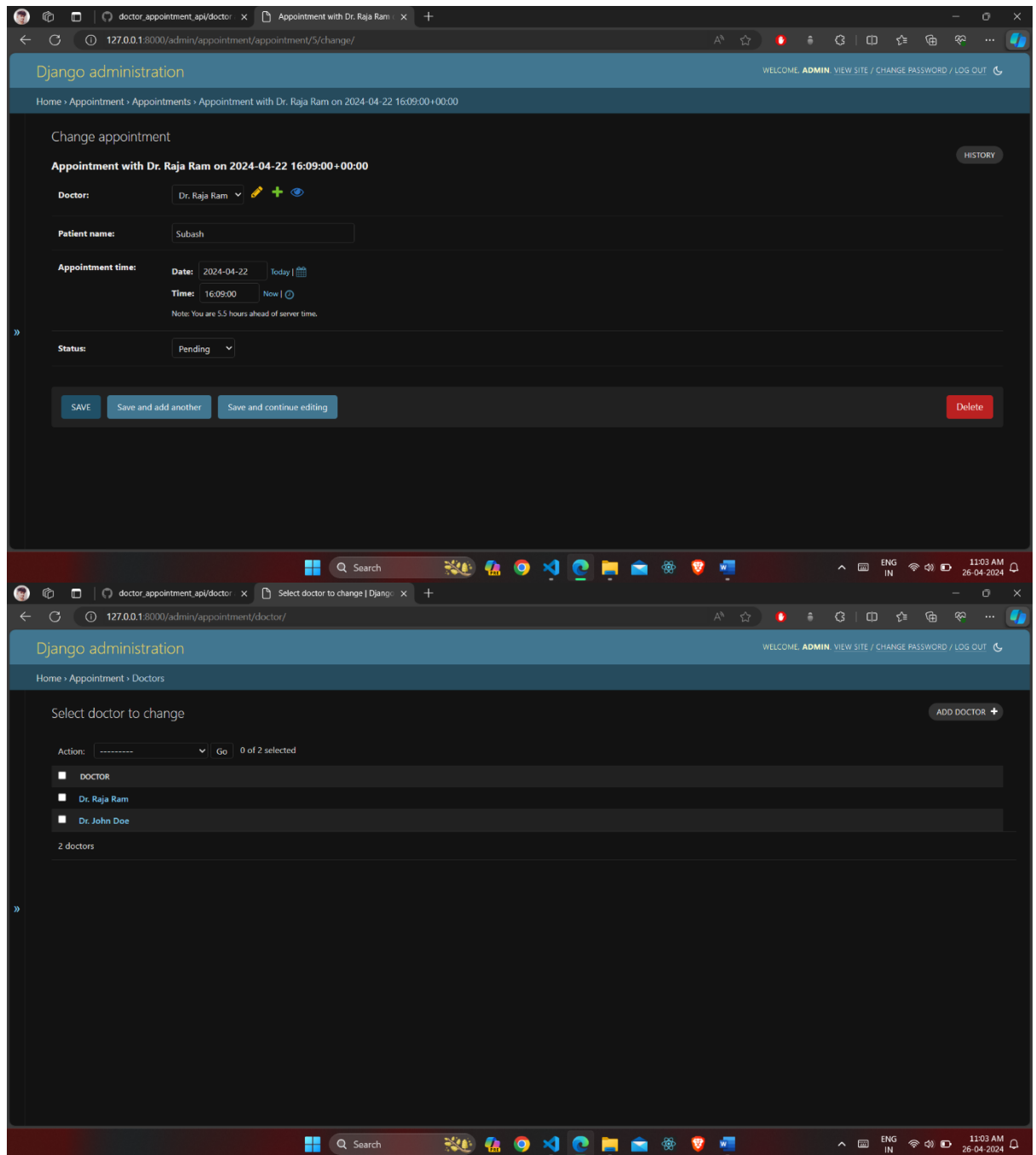


- While using \admin URL it will display the login page of admin panel

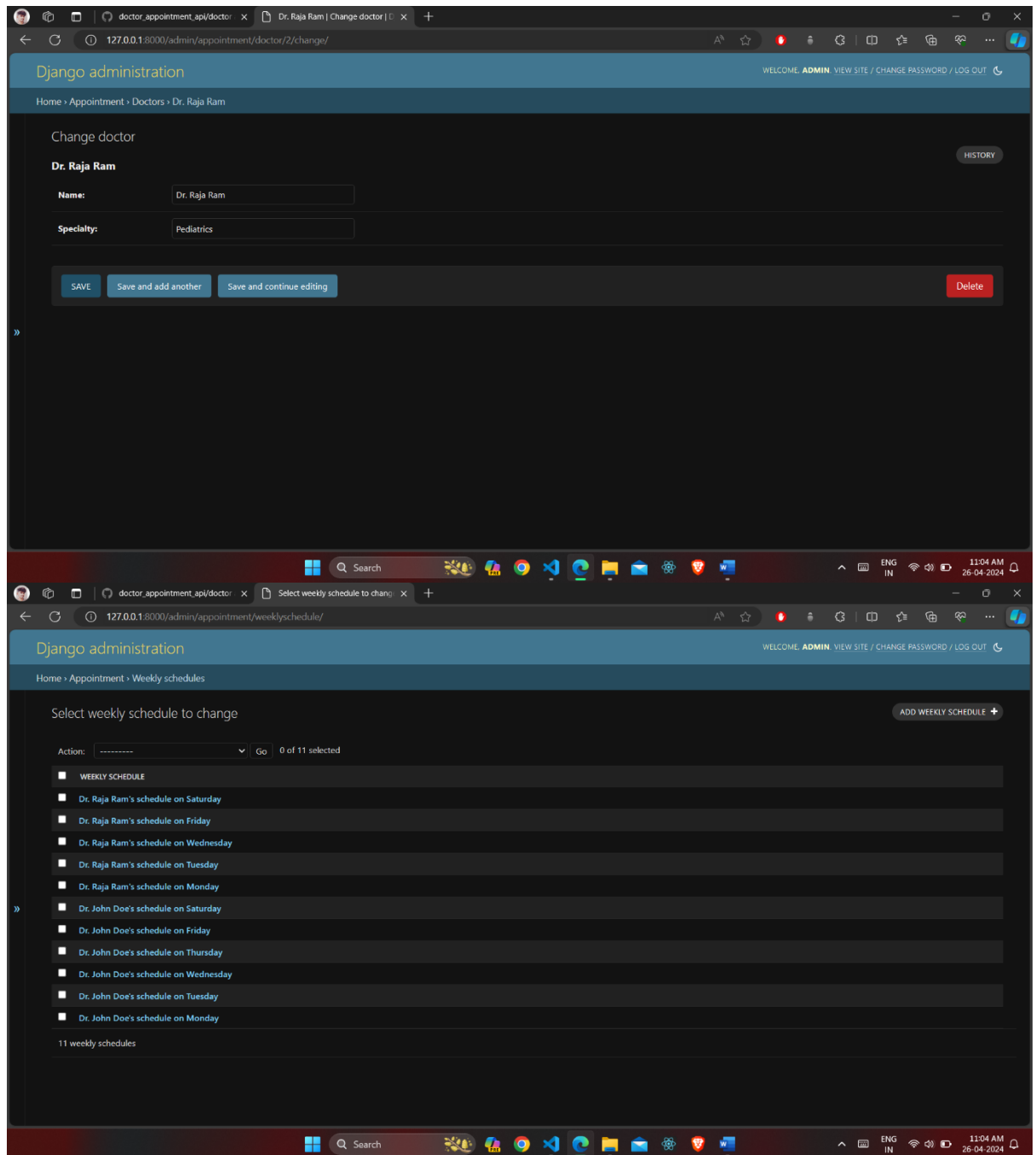
OUTPATIENT APPOINTMENT API



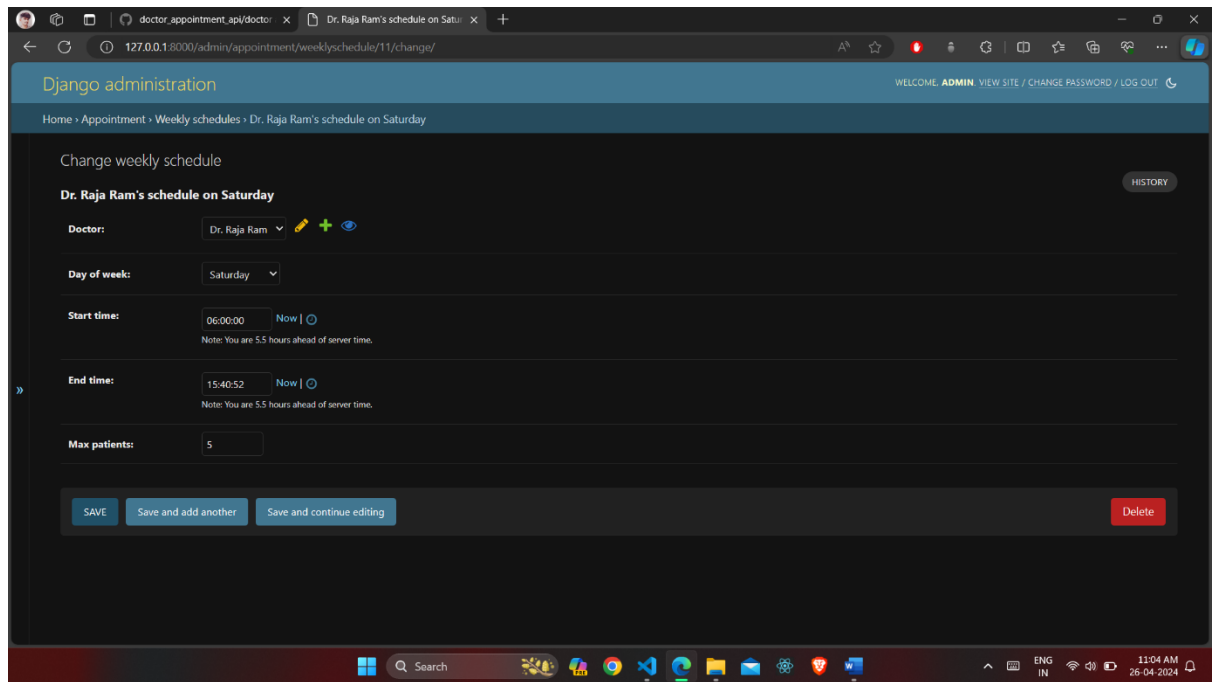
OUTPATIENT APPOINTMENT API



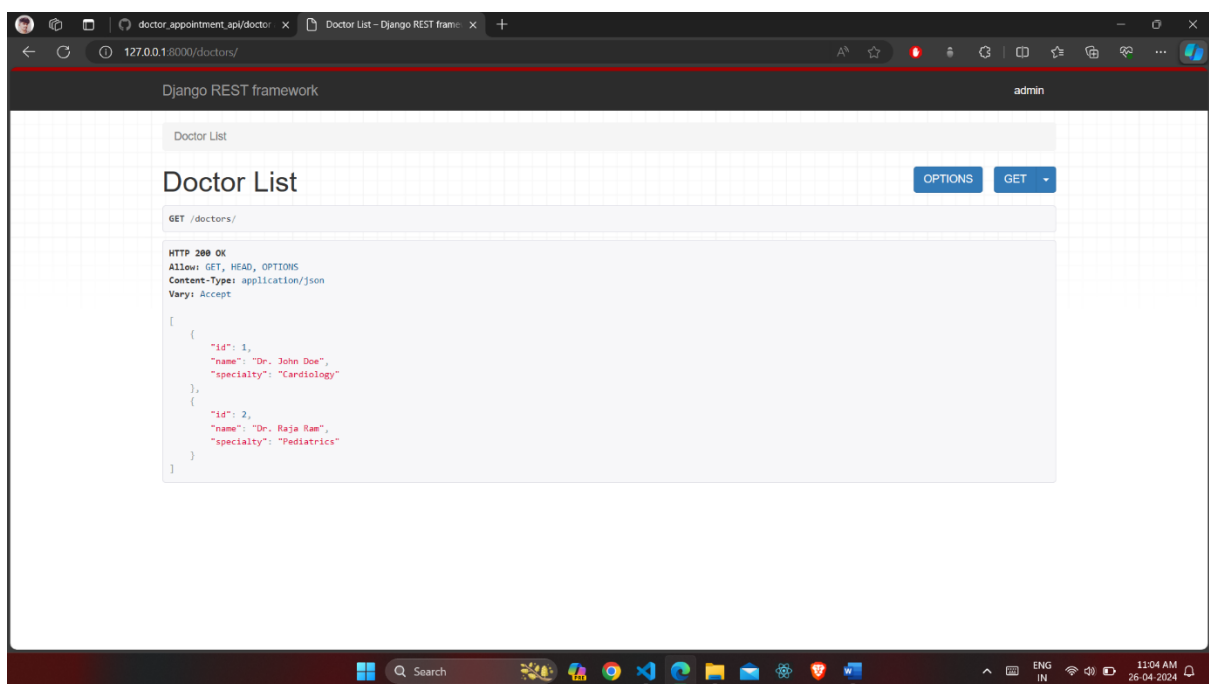
OUTPATIENT APPOINTMENT API



OUTPATIENT APPOINTMENT API

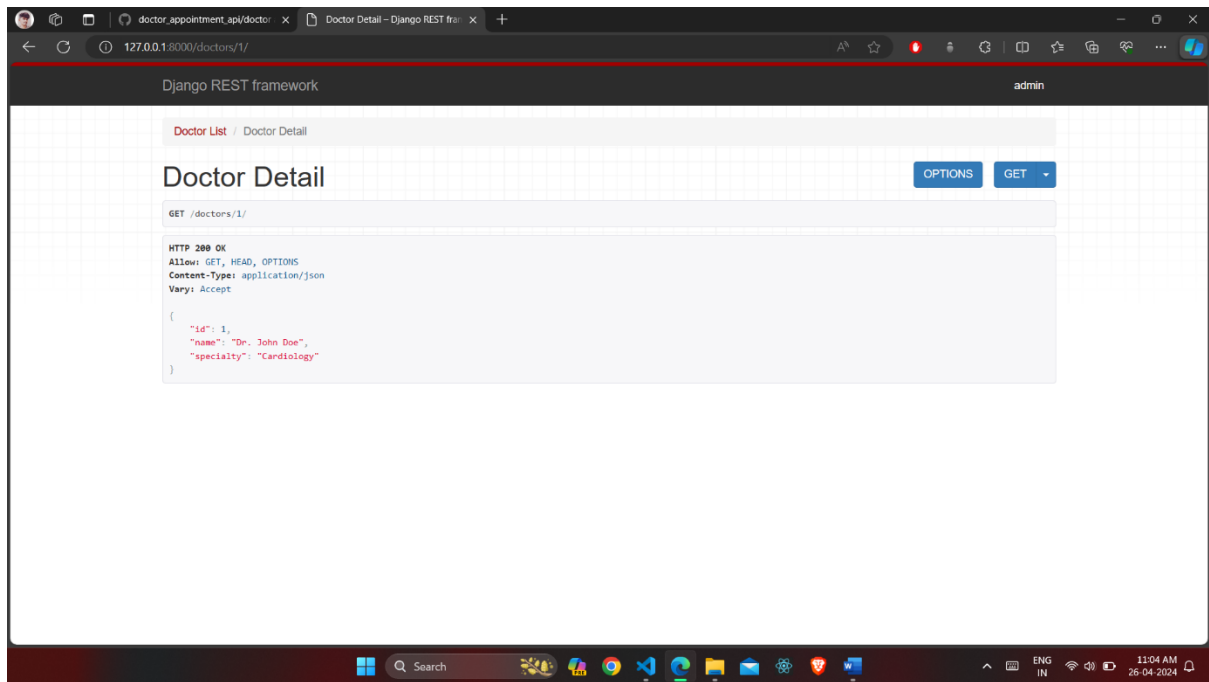


- After login we can able to view the data, edit the data, add the data in the database

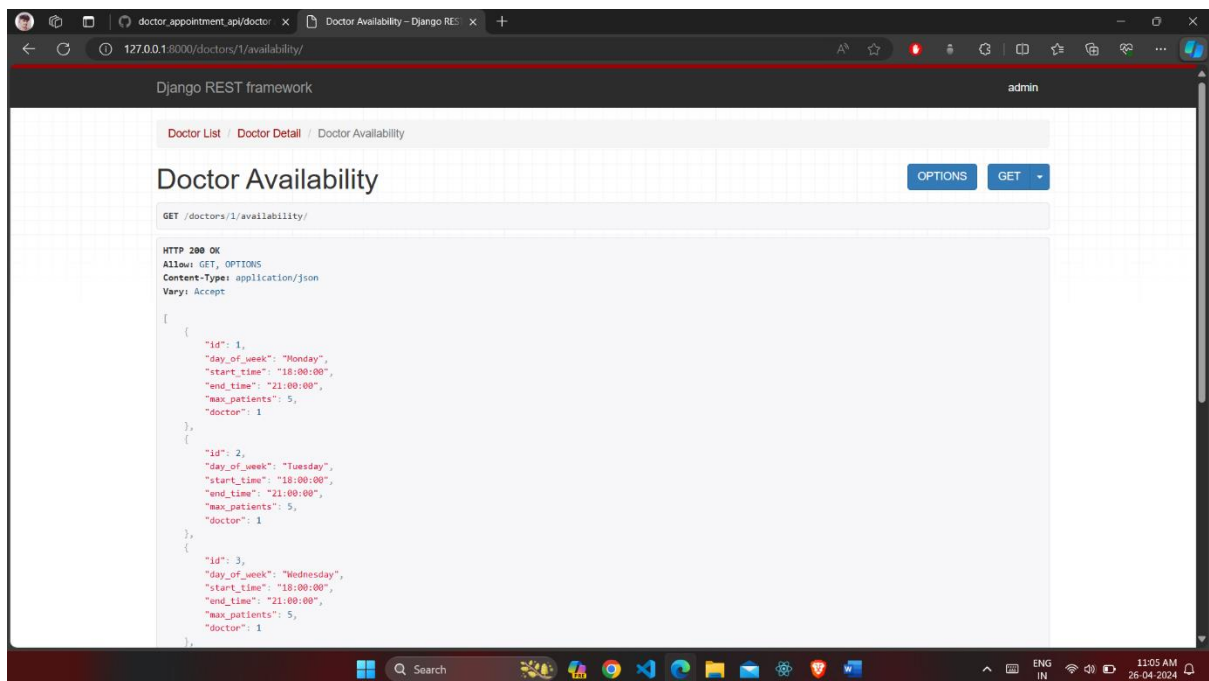


- /doctor URL is used to list the doctors data.

OUTPATIENT APPOINTMENT API

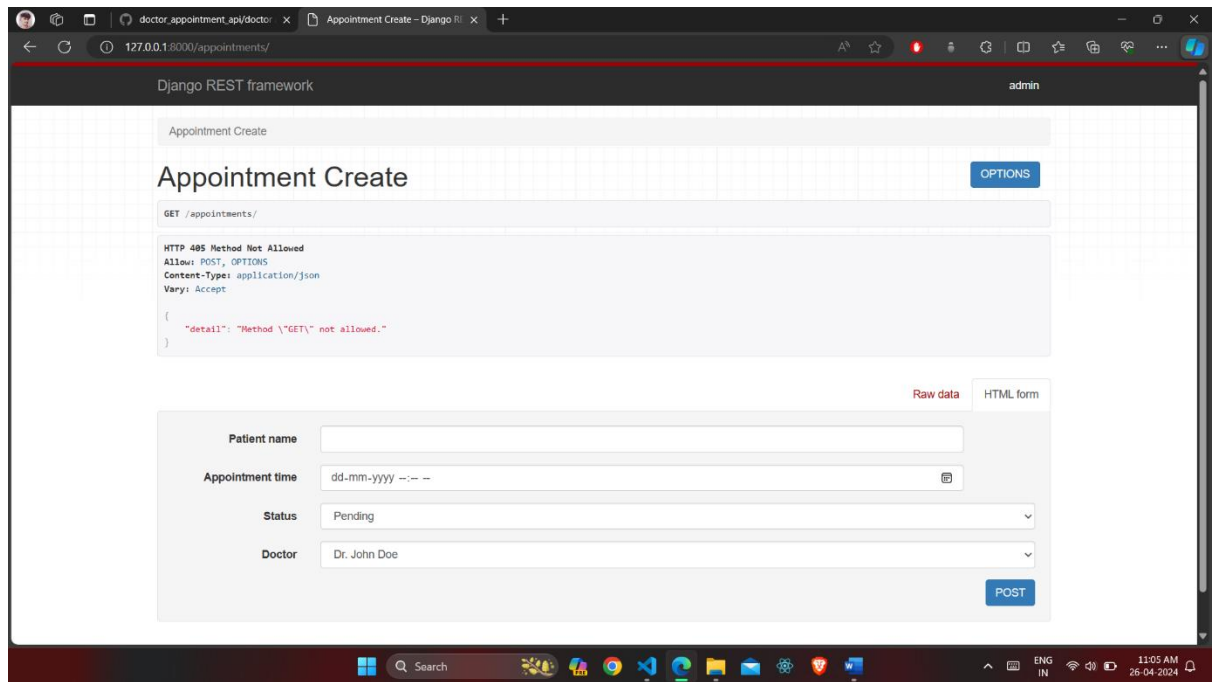


- /doctors/1 or /doctors/2 URL are used to display the particular doctors data by using their id's.



- /doctors/1/availability/ is used to list the availability of the particular doctor in the week.

OUTPATIENT APPOINTMENT API



- /appointments/ is used to register the appointment using post request