4CP33 – FSEP

Python Internship– NSVTech

20CP011 – Jaimin Damor

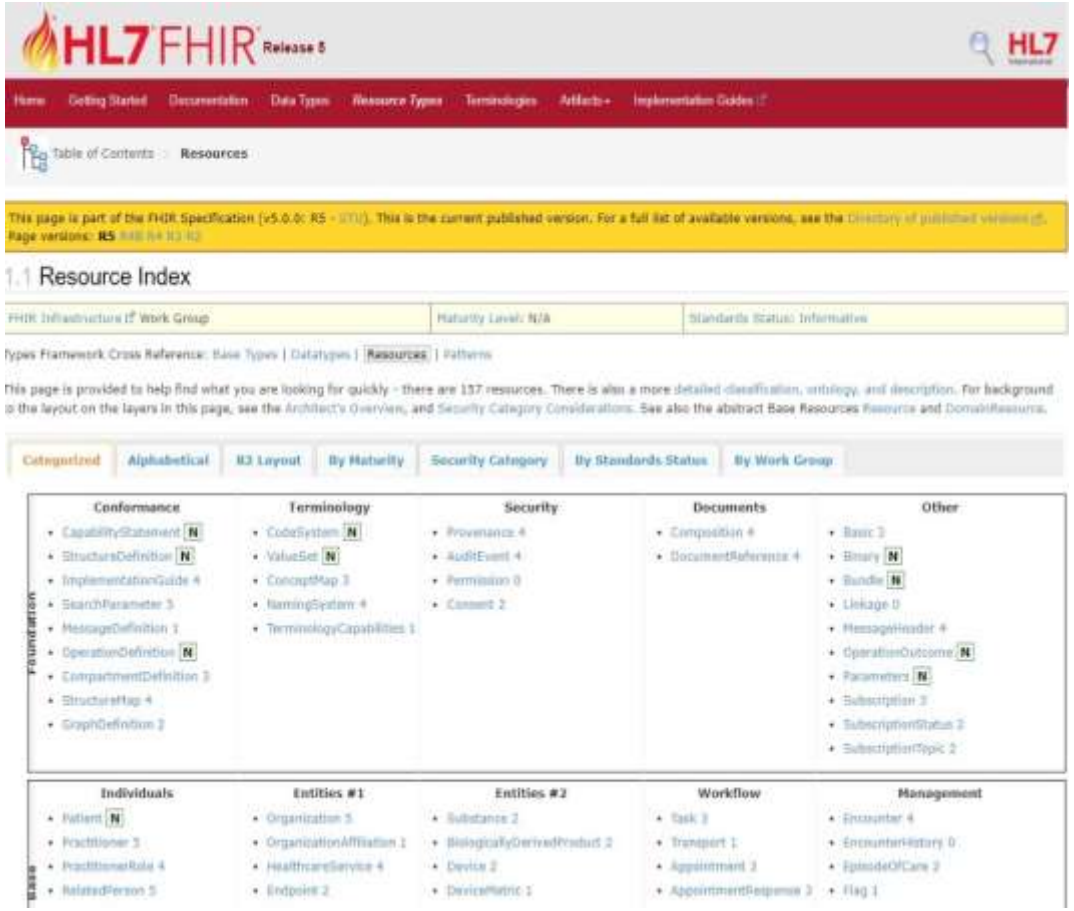
Project_Title : EMR Data Exchange

Build a Electronic Medical Record Application which is a bi-directional interface using the FHIR Standard for seamless sharing and retrieval of electronic medical records (EMRs) among healthcare providers. This initiative ensures secure and standardized transmission of patient data, including demographics, medical history, and treatment plans.:

- Django
- Api’s (Django Rest Framework)
- Creating models (SQLite3-database)
- FHIR Standard

FHIR Standards

It is a Resource Followed by Companies for Exchanging Medical Records of Patients, FHIR is designed to facilitate interoperability, making it easier for different healthcare systems to share and exchange data.



Work Done till now

1st Week:

- HR Induction Program
- Medical Device Demonstration

2nd Week:

- Revise Python language
- End of the week Exam(python)

test9.py

```python
#9. Simulate a simple dice game where two players roll a six-sided die, and the player w
# Use regular expressions to validate the input for player names.
class Bank:
    acc_balance=0
    total_balance_acc_holders=[]

    def __init__(self,name,ac_no):
        self.name=name
        self.ac_no=ac_no

    def deposit(self,ac_no,ammount):
        #self.acc_balance=lambda x:x for i in ammount
        self.acc_balance+=ammount
        print(f"Total balance in your account after deposit :{self.acc_balance}")

    def withdrawal(self,ac_no,ammount):
        self.acc_balance-=ammount
        print(f"Total balance in your account after withdrawal :{self.acc_balance}")

    def check_balance(self):
        print(f"Name : {self.name}")
        print(f"Name : {self.ac_no}")
        print(f"Name : {self.acc_balance}")

    def Transcation_done(self):
        self.total_balance_acc_holders.append(self.acc_balance)

    def display(self):

        x=','.join(map(str,self.total_balance_acc_holders))
        print("Total of all the account_balance :",x)
```

Work Done till now

3rd Week:

- Django Framework
- Working in Virtual Environment
- Create Database
- Field Selection from FHIR standard

Modules

- django.http (sending Response)
- rest_framework.parsers (send data)
- rest_framework.renderers (retrieve data)
- rest_framework.views (API View)
- django.views.decorators.csrf (CSRF token)
- rest_framework.authentication (Valid Auth)
- rest_framework.permissions (Authentication permissions)
- rest_framework.generics (Generic API View)
- rest_framework.routers (router urls)
- rest_framework_simplejwt.views (Token generation/refresh/verification)

Model Creation

Django_project

models.py ● | 0001_initial.py | serializers.py | myapp.py | views.py | urls.py ...\EMRDATA

EMRDATA > EMR_data_exchange > models.py > ...

```python
from django.db import models

# Create your models here.
class Patient(models.Model):
    first_name=models.CharField(max_length=10,null=True,default="")
    last_name=models.CharField(max_length=10,null=True,default="")
    mobile_number=models.CharField(max_length=10,null=True)
    gender=models.CharField(max_length=15,null=True,default="")
    birthdate=models.DateField(blank=True, default='', null=True)
    city=models.CharField(max_length=12,null=True,default="")
    state=models.CharField(max_length=14,null=True,default="")
    pincode=models.CharField(max_length=6,null=True,default="")
    emergency_contant_name=models.CharField(max_length=12,null=True,default="")
    emergency_contant_mobile_number=models.CharField(max_length=10,null=True,default="")
    language=models.CharField(max_length=8,null=True,default="") #null=True,blank=True


    def __str__(self):
        return str(self.mobile_number)


class Procedure(models.Model):
    #USER ,
    patient=models.ForeignKey(Patient, on_delete=models.CASCADE)
    status=models.CharField(max_length=50,default="")
    statusReason=models.CharField(max_length=50,default="")
    category=models.CharField(max_length=50,default="")            # Surgical Procedure
    type=models.CharField(max_length=50,default="")                # Laparoscopic Appendectom
    clinic_address=models.CharField(max_length=50,default="")
    notes=models.CharField(max_length=50,default="")
    report=models.FileField(null=True,blank=True)
```

Call functions

myapp.py | views.py ✕ | urls.py ...\EMRDATA | tests.py | urls.py ...\EMR

EMRDATA > EMR_data_exchange > views.py > patientinfo

```python
from .models import Patient,Procedure
from .serializers import PatientSerializer
from rest_framework.parsers import JSONParser
from rest_framework.renderers import JSONRenderer
Create your views here.
def patientinfo(request):
    if request.method=='GET':
        json_data=request.body
        stream=io.BytesIO(json_data)
        pythondata=JSONParser().parse(stream)
        id=pythondata.get('id',None)
        if id is not None:
            pat=Patient.object().get(id=id)
            serializer=PatientSerializer(pat)
            json_data=JSONRenderer().render(serializer)
            return HttpResponse(json_data,content_type='application/text')

        Pat=Patient.objects.all()
        serializer=PatientSerializer(Pat)
        json_data=JSONRenderer.render(serializer.data)
        return HttpResponse(json_data,content_type='application/text')
```

Migrations in DB

models.py | db.sqlite3 | 0001_initial.py | serializers.py | myapp.py | views.py | urls.py ...\EMRDATA

EMRDATA > db.sqlite3

Search tables... | Reset Filters | Records: 20 | Search 20 recor

Tables (13)

- django_migrations
- sqlite_sequence
- EMR_data_exchange_procedure
- auth_group_permissions
- auth_user_groups
- auth_user_user_permissions
- django_admin_log
- django_content_type
- auth_permission
- auth_group
- auth_user
- django_session
- EMR_data_exchange_patient

	id	app	name	applied
1	1	EMR_data_exchange	0001_initial	2024-02-02 10:27:52....
2	2	contenttypes	0001_initial	2024-02-02 10:27:52....
3	3	auth	0001_initial	2024-02-02 10:27:52....
4	4	admin	0001_initial	2024-02-02 10:27:52....
5	5	admin	0002_logentry_remov...	2024-02-02 10:27:52....
6	6	admin	0003_logentry_add_a...	2024-02-02 10:27:52....
7	7	contenttypes	0002_remove_conten...	2024-02-02 10:27:52....
8	8	auth	0002_alter_permissio...	2024-02-02 10:27:52....
9	9	auth	0003_alter_user_emai...	2024-02-02 10:27:52....
10	10	auth	0004_alter_user_user...	2024-02-02 10:27:52....
11	11	auth	0005_alter_user_last_l...	2024-02-02 10:27:52....
12	12	auth	0006_require_content...	2024-02-02 10:27:52....
13	13	auth	0007_alter_validators...	2024-02-02 10:27:52....
14	14	auth	0008_alter_user_user...	2024-02-02 10:27:52....
15	15	auth	0009_alter_user_last_...	2024-02-02 10:27:52....
16	16	auth	0010_alter_group_na...	2024-02-02 10:27:52....
17	17	auth	0011_update_proxy_...	2024-02-02 10:27:52....
18	18	auth	0012_alter_user_first_...	2024-02-02 10:27:52....

Reading through Api

myapp.py | views.py | urls.py ...\EMRDATA | tests

EMRDATA > myapp.py > get_data > id

```python
import requests
import json

URL="http://127.0.0.1:8000/patientinfo/"

def get_data(id=None):
    data={}
    if id is not None:
        data={'id':id}
        print(id)
        json_data=json.dumps(data)
        r=requests.get(url=URL,data=json_data)
        data=r.json()
        print(data)


get_data(1)
```

Class Based View

EMRDATA

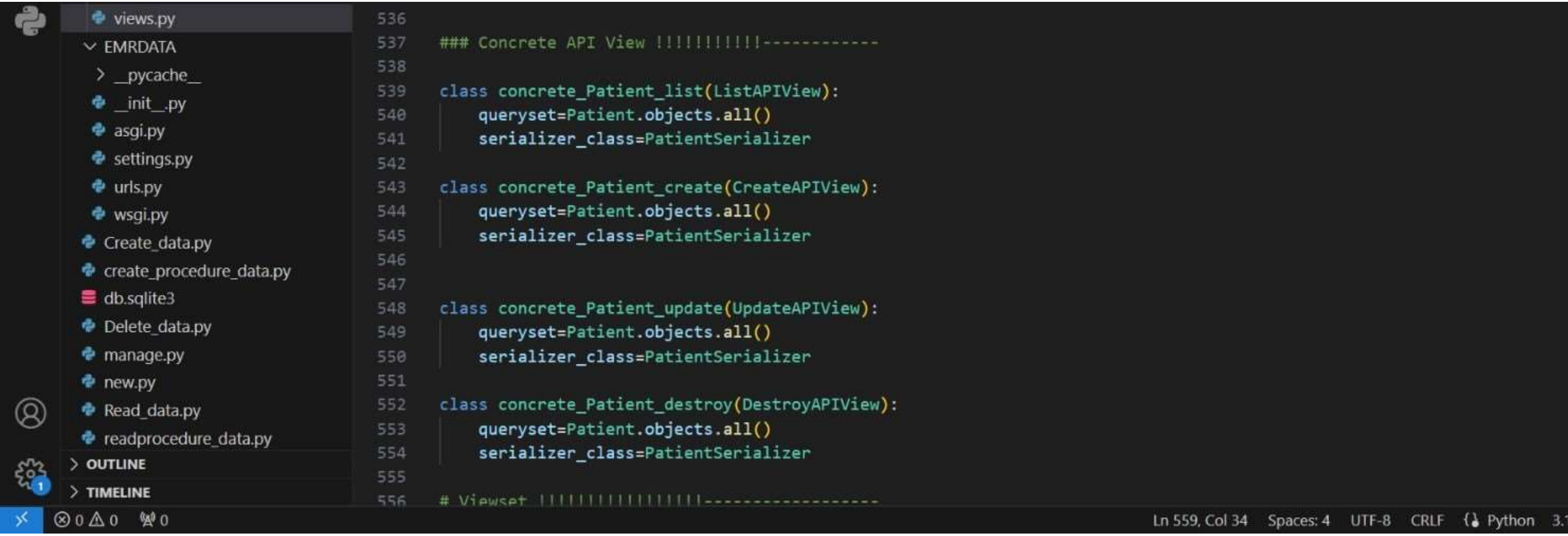
py | views.py | settings.py | Create_data.py | serializers.py | custompremission.py

ata_exchange > views.py > PatientViewSet

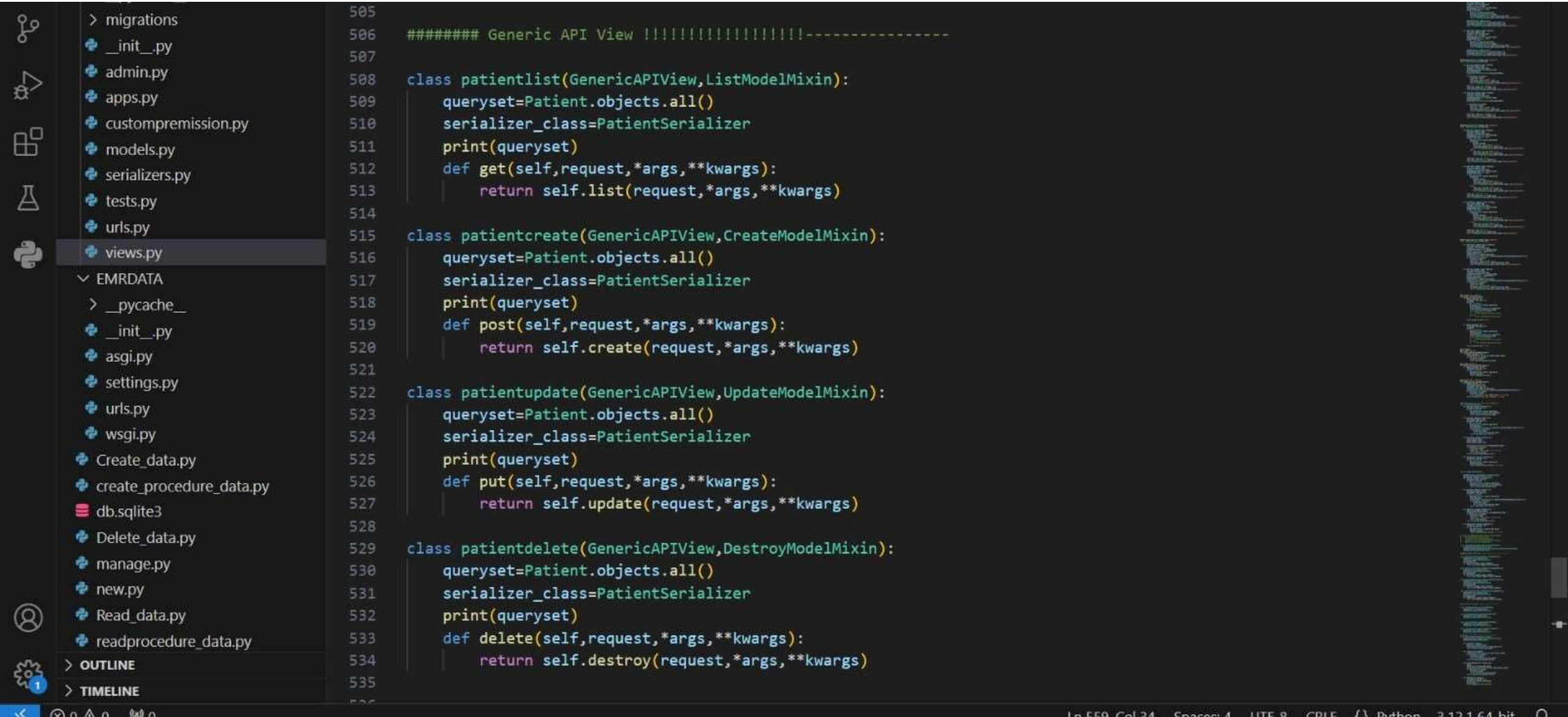
```python
# CLASS based API''''''ssssssssssssssssssss !!!!!!!!!!!!!!!!!!!!!


####### Procedure API''s
@method_decorator(csrf_exempt,name='dispatch')
class update_procedure_dataAPI(View):

    def patch(self,request,*args,**kwargs):
        json_data=request.body
        stream=io.BytesIO(json_data)
        pythondata=JSONParser().parse(stream)
        id=pythondata.get('id',None)

        procedure_data=Procedure.objects.get(id=id)
        serializer=ProcedureSerializer(instance=procedure_data,data=pythondata,partial=True)
        if serializer.is_valid():
            serializer.save()
            res={'msg':' Data Updated !!!!!!!!!'}
            json_data=JSONRenderer().render(res)
            return HttpResponse(json_data,content_type='application/text')

    def put(self,request,*args,**kwargs):
        json_data=request.body
        stream=io.BytesIO(json_data)
        pythondata=JSONParser().parse(stream)
        id=pythondata.get('id',None)

        procedure_data=Procedure.objects.get(id=id)
        serializer=ProcedureSerializer(instance=procedure_data,data=pythondata,partial=True)
        if serializer.is_valid():
            serializer.save()
```

Ln 559, Col 34 Spaces: 4 UTF-8 CR

Concrete Based View



Generic View



Normal URL's

```
path('patientlist/',views.patientlist.as_view(),name='patientlist'),
path('patientcreate/',views.patientcreate.as_view(),name='patientcreate'),
path('patientupdate/<int:pk>',views.patientupdate.as_view(),name='patientupdate'),
path('patientdelete/<int:pk>',views.patientdelete.as_view(),name='patientdelete'),
```

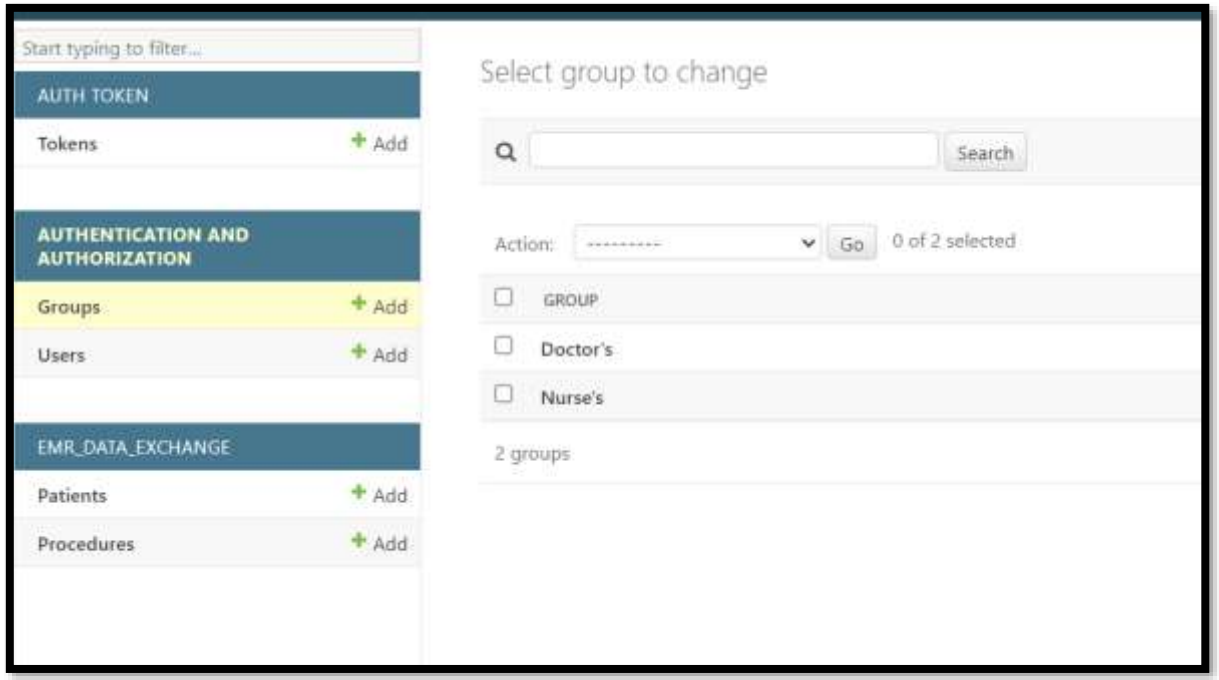
Router URL's

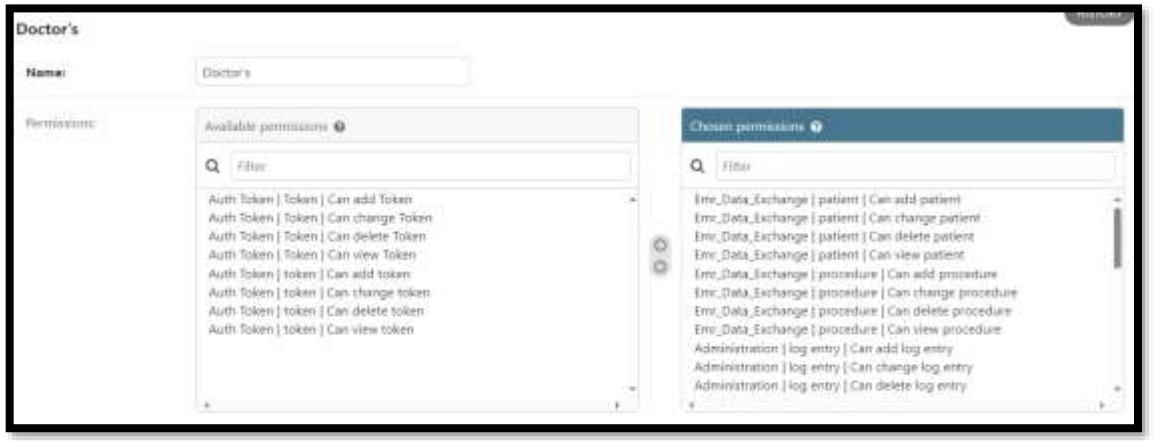
```
from rest_framework.routers import  DefaultRouter
router= DefaultRouter()

router.register('PatientViewSet',views.PatientViewSet,basename="doctor")
router.register('PatientModelViewSet',views.PatientModelViewSet,basename="None")
```

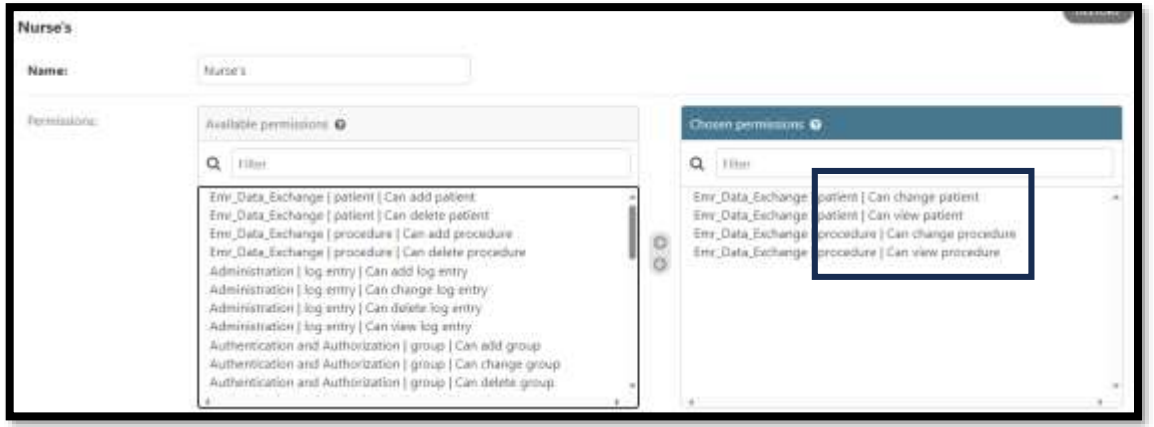
```
def patientcreate(self,request):
    serializer=PatientSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors)

def patientupdate(self,request,pk):
    id=pk
    pat=Patient.objects.get(id=id)
    serializer=PatientSerializer(pat,data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors)
```

PUT http://127.0.0.1:8000/patientupdate/14

Authentication/Authorization



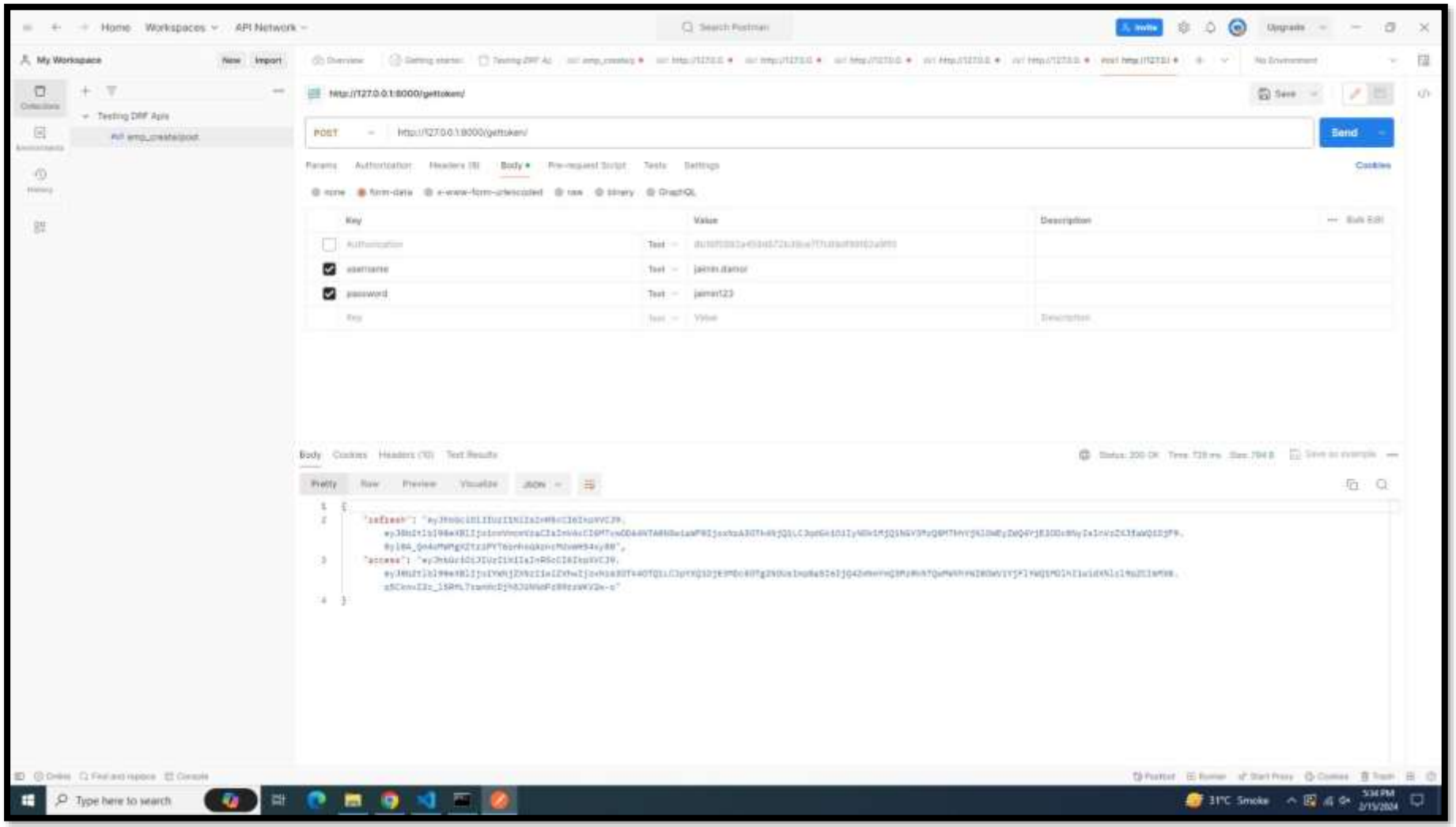




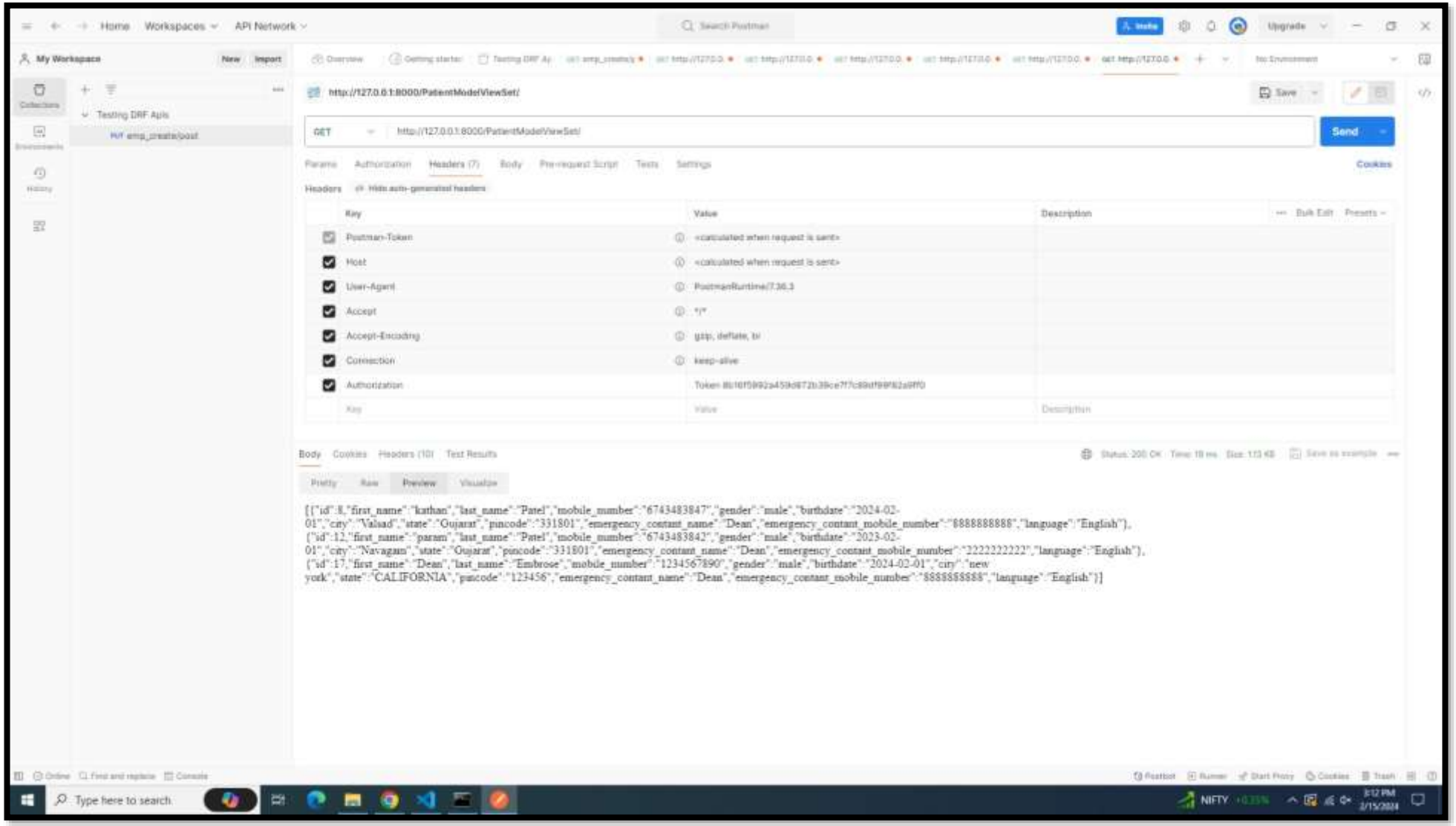
Viewset View

```python
# Viewset !!!!!!!!!!!!!!!!!!----------------
class PatientViewSet(viewsets.ViewSet):
    authentication_classes=[SessionAuthentication]
    permission_class=[IsAdminUser]          #IsAuthenticated,AllowAny,DjangoModelPermissions

    def list(self,request):
        pat=Patient.objects.all()
        serializer=PatientSerializer(pat,many=True)
        return Response(serializer.data)

    def create(self,request):
        serializer=PatientSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors)

    def partialupdate(self,request,pk):
        id=pk
        pat=Patient.objects.get(id=id)
        serializer=PatientSerializer(pat,data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors)

    def delete(self,request):
        id=request.data.get('id')
        pat=Patient.objects.get(id=id)
        pat.delete()
        return Response("Data Deleted")
```

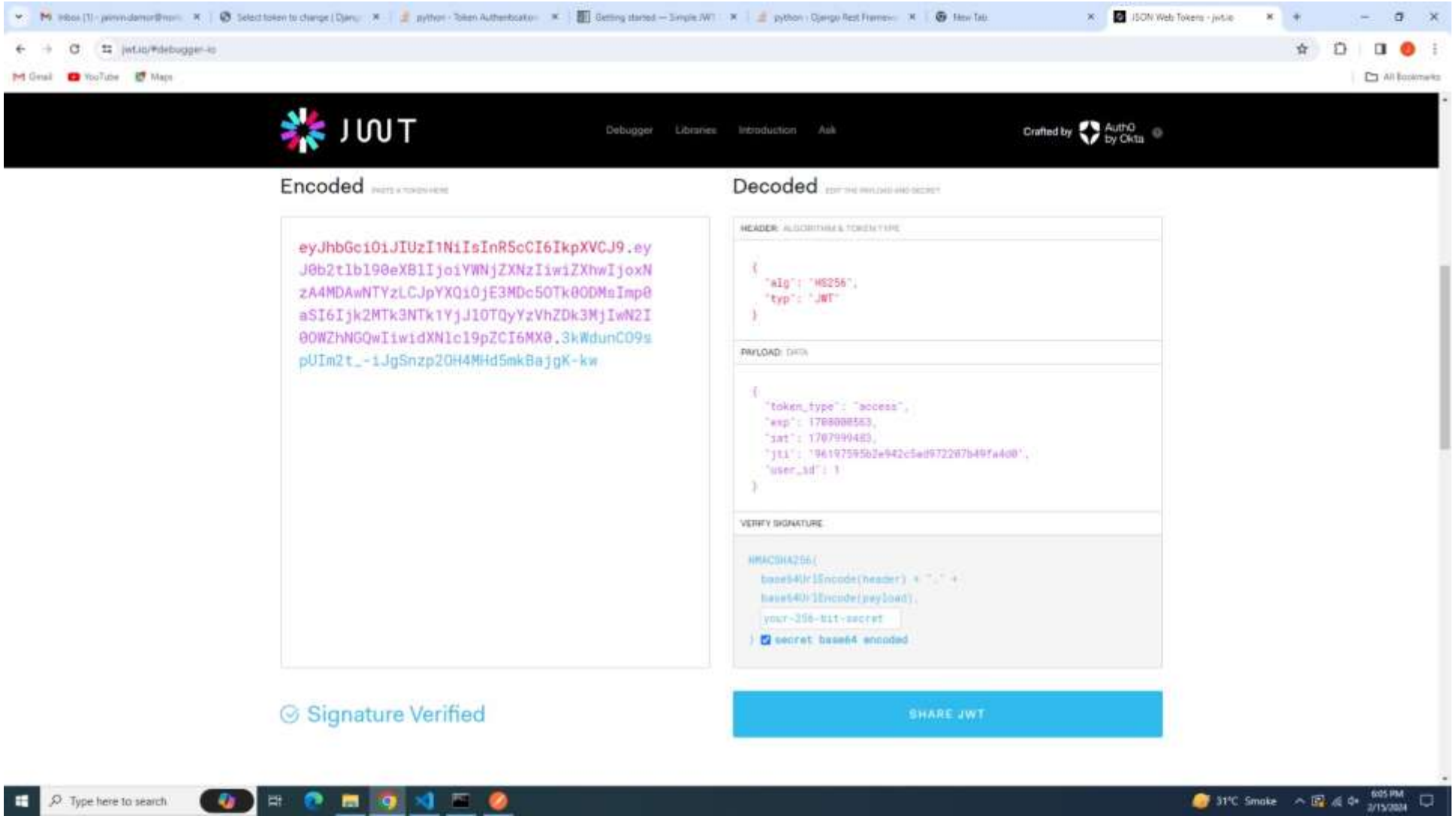
Token generation



Retrieve Data (Token Auth)



Token Verification



THANK YOU