1. Models.py:

# Create your models here.

from django.db import models

class Frequency(models.Model):

    value = models.FloatField()

Frequency is class name, it contains one float field(frequency value)

1. Serializable.py (create it on own inside sineapp)

from rest\_framework import serializers

from .models import Frequency

class FrequencySerializer(serializers.ModelSerializer):

    class Meta:

        model = Frequency

        fields = '\_\_all\_\_'

serializer used to convert model instance to json and viceversa

1. Urls.py

from django.urls import include, path

from rest\_framework import routers

from sineapp import views

router = routers.DefaultRouter()

router.register(r'frequencies', views.FrequencyViewSet)

urlpatterns = [

    path('', include(router.urls)),

]

1. Views.py

from django.shortcuts import render

import matplotlib.pyplot as plt

# Create your views here.

from rest\_framework import viewsets

from .models import Frequency

from .serializers import FrequencySerializer

from rest\_framework.decorators import action

from rest\_framework.response import Response

from django.http import FileResponse

from io import BytesIO

from rest\_framework.decorators import action

from rest\_framework.response import Response

import numpy as np

import matplotlib.pyplot as plt

class FrequencyViewSet(viewsets.ModelViewSet):

    queryset = Frequency.objects.all()

    serializer\_class = FrequencySerializer

    @action(detail=False, methods=['GET'])

    def plot\_frequencies(self, request):

        frequencies = Frequency.objects.all()

        values = [frequency.value for frequency in frequencies]

        # Generate x values from 0 to 2\*pi with the same length as the frequency values

        x = np.linspace(0, 2\*np.pi, len(values))

        # Generate y values using the frequency values as the amplitude of the sine wave

        y = np.sin(values \* x)

        # Plot the sine wave

        plt.plot(x, y)

        plt.xlabel('X')

        plt.ylabel('Y')

        plt.title('Sine Wave Plot')

        plt.show()

        return Response({'message': 'Sine wave plot generated'})

5. Settings.py

from pathlib import Path

import os

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-tv=!r0o9q8h7vrr()&q)^dudjgij$0-+oj5l@wqhqr(3imhv$g'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

    'sineapp',

    'rest\_framework',

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

]

MIDDLEWARE = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'sinewave\_generator.urls'

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

WSGI\_APPLICATION = 'sinewave\_generator.wsgi.application'

# Database

# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

'''DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}'''

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

    }

}

# Password validation

# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

# Internationalization

# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC\_URL = 'static/'

# Default primary key field type

# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

Under installed apps include sineapp and restframework

Make changes in database also.













