1. *Search Implementation through URL :*

First, you need to install the Django Filter package if you haven't already:

```
pip install django-filter
```

Update your urls.py to include a new URL pattern that accepts the search parameters as path parameters:

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
from rest_framework import routers
from coins.views import CoinViewSet, CoinSearchView
# Create a router for registering viewsets
router = routers.DefaultRouter()
# Register CoinViewSet with the router
router.register(r'coins', CoinViewSet)
# Define URL patterns
urlpatterns = [
    # Admin site URL
    path('admin/', admin.site.urls),
    # API endpoints for coins using the router
    path('api/', include(router.urls)),
    path('coins/search/<path:path_params>/', CoinSearchView.as_view(), name='coin-search'),
]
# Serve media files in DEBUG mode
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

In this updated URL pattern, <str:field> represents the field name (e.g., coin_name, coin_desc, coin_year), and <str:value> represents the corresponding search value.

To handle a variable number of fields and values in the URL, you can modify the view to parse the URL dynamically. You can use regular expressions to capture the field-value pairs from the URL and then filter the queryset accordingly :

```
from django.shortcuts import render  # Import render function from Django

from rest_framework import status  # Import status codes from Django REST Framework

from rest_framework.response import Response  # Import Response class from Django REST Framework

from rest_framework import viewsets  # Import viewsets from Django REST Framework

from .models import Coin  # Import the Coin model

from .serializers import CoinSerializer  # Import the CoinSerializer

from rest_framework.views import APIView

import re

class CoinViewSet(viewsets.ModelViewSet):  # Define a viewset for the Coin model

    queryset = Coin.objects.all()  # Define the queryset to fetch all coin objects

    serializer_class = CoinSerializer  # Specify the serializer class to use for the Coin model

class CoinSearchView(APIView):

    def get(self, request, *args, **kwargs):

        # Extract search parameters from the URL path

        path_params = kwargs.get('path_params')

        # Parse the path_params string into field-value pairs

        search_params = {}

        if path_params:

            # Split the path_params string by '/'

            path_params_list = path_params.split('/')

            # Ensure there are an even number of elements (field-value pairs)
```

```
        if len(path_params_list) % 2 == 0:
            for i in range(0, len(path_params_list), 2):
                search_params[path_params_list[i]] = path_params_list[i+1]
    # Filter Coin objects based on the provided search parameters
    coins = Coin.objects.all()
    for field, value in search_params.items():
        coins = coins.filter(**{field: value})
    if not coins:
        return Response({"message": "No coins found for the provided search parameters"}, status=404)
    # Serialize the filtered queryset
    serializer = CoinSerializer(coins, many=True, context={'request': request})
    return Response(serializer.data)
```

This setup allows you to construct URLs with a flexible number of field-value pairs, enabling dynamic searches based on your preferences.

==========