Python Code File: google-auth/asgi.py

```python
# ASGI config for googleauth project.
# It exposes the ASGI callable as a module-level variable named application.
import os
from django.core.asgi import get_asgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'googleauth.settings')
application = get_asgi_application()
```

Python Code File: google-auth/wsgi.py

```python
# WSGI config for googleauth project.
# It exposes the WSGI callable as a module-level variable named application.
import os
from django.core.wsgi import get_wsgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'googleauth.settings')
application = get_wsgi_application()
```

Python Code File: google-auth/urls.py

```python
# Django settings for googleauth project.
# Generated by 'django-admin startproject' using Django 3.2.8.
from django.contrib import admin
from django.urls import base, path,include
from django.views.generic import TemplateView
from API import views
import base_app

urlpatterns = [
    path('', TemplateView.as_view(template_name="base/index.html")),
    path('admin/', admin.site.urls),
    path('accounts/', include('allauth.urls')),
    path('cluster/', include('base_app.urls')),
    path('api/', include('API.urls')),
    path('paypal/', include('paypal.standard.ipn.urls')),
    path('payment/', base_app.views.testpayment, name='payment'),
]
```

Python Code File: google-auth/settings.py

```python
# Django settings for googleauth project.
# Generated by 'django-admin startproject' using Django 3.2.8.
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/3.2/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-+&9n$5zghxvddg3*1i!f-@o53-8jt#z0b-!+hefv*)e@@8*f_*'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = ['*']

# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Django-allauth
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'allauth.socialaccount.providers.google',

    # Django-crispy-forms
    'crispy_forms',
    'base_app.apps.BaseAppConfig',

    # Django-elasticsearch-dsl
    'django_elasticsearch_dsl',
    'rest_framework',
    'rest_framework.authtoken',
    'API',

    # Payment gateway
    'paypal.standard.ipn',
]
```

```
ELASTICSEARCH_DSL={
    'default': {
        'hosts': 'localhost:9200'
    },
}
SITE_ID = 1
LOGIN_REDIRECT_URL = "/cluster"
ACCOUNT_LOGOUT_REDIRECT_URL ="/accounts/login"
LOGIN_URL = "/accounts/login"
# ACCOUNT_EMAIL_UNIQUE = True

# Authentication features which enables password reset and verifies registered email
ACCOUNT_AUTHENTICATION_METHOD = 'email'
ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_EMAIL_VERIFICATION = 'mandatory'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_USE_TLS = True
EMAIL_PORT = 587
EMAIL_HOST_USER = 'dummyCseProject@gmail.com'
EMAIL_HOST_PASSWORD = 'Cse327dummy'

# Django-Paypal settings
PAYPAL_RECEIVER_EMAIL = 'dummyCseProject@gmail.com'
PAYPAL_TEST = True
# Business account
# Email: sb-igibe15768129@business.example.com
# Password: O]E6S7_n
# Personal account
# Email: sb-aoso315701263@personal.example.com
# Password: 3:Vr,O(.

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 = 'googleauth.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        '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 = 'googleauth.wsgi.application'
# Database
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        #'NAME': BASE_DIR / 'db.sqlite3',
        'NAME': str(os.path.join(BASE_DIR, 'db.sqlite3')),
    }
}
AUTHENTICATION_BACKENDS = (
    # Needed to login by username in Django admin, regardless of `allauth`
    'django.contrib.auth.backends.ModelBackend',
    # `allauth` specific authentication methods, such as login by e-mail
    'allauth.account.auth_backends.AuthenticationBackend',
)
# Password validation
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/3.2/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
STATIC_URL = '/static/'
# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
# Django-crispy-forms
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

Python Code File: base_app/admin.py

```
from django.contrib import admin
from .models import Cluster, Url
# Register your models here.
admin.site.register(Cluster)
admin.site.register(Url)
```

Python Code File: base_app/apps.py

```
from django.apps import AppConfig
class BaseAppConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'base_app'

    def ready(self):
        from base_app import updater
        updater.start()
```

Python Code File:base_app/es_paginator.py

```
from django.utils.functional import LazyObject
class SearchResults(LazyObject):
    def __init__(self, search_object):
        self._wrapped = search_object

    def __len__(self):
        return self._wrapped.count()
```

```
    def __getitem__(self, index):
        search_results = self._wrapped[index]
        if isinstance(index, slice):
            search_results = list(search_results)
        return search_results

Python Code File:base_app/export.py
 import json
 import requests
 from bs4 import BeautifulSoup
 import csv
 def link(URL):
    print('exporting', URL)
    i=0
    if URL is not None:
       r = requests.get(URL)
       print(URL)
       soup = BeautifulSoup(r.content, 'lxml')
       table = soup.find_all('td', class_='lin')
       f = csv.writer(open('D:\Test folder (2)\demo1.csv', 'w'))
       f.writerow(["Links"])

       for t in table:
           print(t)
           print(i)
           i+=1
           li = t.a.text
           f.writerow([li])
       print('exported!!!!!!!!!!!!!!!!!!')
   print("finished")

Python Code File: base_app/forms.py
 from django import forms
 from .models import Cluster, Url
 class UrlForm(forms.ModelForm):
    class Meta:
       model = Url
       fields = ('cluster','cluster_url', 'depth', 'output_type')

    def __init__(self, cluster_id=None, *args, **kwargs):
       super().__init__(*args, **kwargs)
```

```
        if cluster_id:
            self.fields['cluster'].initial = cluster_id
            self.fields['cluster'].widget = forms.HiddenInput()

Python Code File: base_app/models.py
 from django.conf.urls import url
 from django.db import models
 from django.contrib.auth.models import User
 from django.template.defaultfilters import slugify
 from spider.spider.spiders.crawler import begin_crawl

 # Cluster
 # One user can have multiple clusters
 # An one-to-many relationship has been created
 class Cluster(models.Model):
    # Once the user deletes his account,
    # Clusters related to his account will be deleted
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True,
blank=True, related_name='clusters')
    title = models.CharField(max_length=100)
    description = models.TextField(null=True, blank=True, default='Creating a
Search Cluster for advanced search')
    date_created = models.DateTimeField(auto_now_add=True)
    date_updated = models.DateTimeField(auto_now=True)
    slug = models.SlugField(null=True, blank=True)

    class Meta:
        # Default ordering is set to date_created
        ordering = ['date_created']
    def __str__(self):
        return self.title
    def save(self, *args, **kwargs):
        self.slug = slugify(self.title)
        super(Cluster, self).save(*args, **kwargs)

 # Each Cluster can have multiple urls
 # An one-to-many-relationship has been created
 output_preference = (
        ('html', 'Textual Data'),
        ('pdf', 'PDF File'),
        ('txt', 'TXT File'),
```

```
        ('doc', 'DOC File'),
        ('docx', 'Docx File'),
    )
class Url(models.Model):
    # Once a Cluster is deleted,
    # All the urls related to that cluster will be deleted
    cluster = models.ForeignKey(Cluster, on_delete=models.CASCADE, null=True,
blank=True, related_name='urls')
    cluster_url = models.URLField(max_length=200)
    depth = models.PositiveIntegerField()
    output_type = models.CharField(max_length=20, choices=output_preference)
    date_added = models.DateTimeField(auto_now_add=True)
    slug = models.SlugField(null=True, blank=True)
    is_crawled= models.BooleanField(default=False)

    class Meta:
        # Default ordering is set to date_created
        ordering = ['date_added']
    def __str__(self):
        return self.cluster_url
    def save(self, *args, **kwargs):
        self.slug = slugify(self.cluster)
        super(Url, self).save(*args, **kwargs)

Python Code File: base_app/schedule.py
 from .models import Url, Cluster
 from .scrape import crawl
 from .sendMail import send

 def scheduled_crawl():
    print('Testing Crawl Status!!!!!!!!!!!!!!!!!!')
    uncrawled_url_users=set()
    urls = Url.objects.all()

    for url in urls.iterator():
        current_url = Url.objects.get(id=url.id)
        cluster_id = current_url.cluster.id
        link= url.cluster_url
        height= url.depth
        type= url.output_type
```

```
        #if the url is not crawled then crawl and update the status
        if current_url.is_crawled is not True:
            uncrawled_url_users.add(url.cluster.user.email)
            #uncrawled_url_users = uncrawled_url_users.append(cluster_id.user.id)
            #print(uncrawled_url_users)
            crawled = crawl(link, height, type, cluster_id)

            if crawled:
                current_url.is_crawled = crawled
                current_url.save()
                print(link)
                print(current_url.is_crawled)
            else:
                print('Error occured whiled crawling!!!!!!!')
        else:
            print('url: ' + str(current_url.id) + ' already crawled.......')
    for mail in uncrawled_url_users:
        send(mail)
    uncrawled_url_users.clear()
    print('Ended crawling')

Python Code File: base_app/scrape.py
 # Import libraries
 from urllib.request import urljoin
 from bs4 import BeautifulSoup
 from bs4.builder import TreeBuilder
 import requests
 from urllib.request import urlparse
 from elasticsearch import Elasticsearch
 import time
 import PyPDF2
 import os
 import requests
 from urllib.parse import urljoin
 from bs4 import BeautifulSoup
 import textract
 import unicodedata
 import re
 import itertools
```

```
import sys
from itertools import chain
# Method for crawling a url at next level
def level_crawler(input_url, depth, type, cluster_id):
   links_extern = set()
   control_chars = ''.join(
              map(chr, chain(range(0, 9),range(11, 32),range(127, 160))))
   CONTROL_CHAR_RE = re.compile('[%s]' % re.escape(control_chars))
   text = ''
   links_intern = set()
   temp_urls = set()
   current_url_domain = urlparse(input_url).netloc
   # Beautiful soup object to extract html tags
   beautiful_soup_object = BeautifulSoup(
       requests.get(input_url).content, "lxml")
   if(type == 'html'):
       for script in beautiful_soup_object(['style', 'script',
                               '[document]', 'head', 'title']):
           script.extract()
       text = beautiful_soup_object.get_text(strip=True)
       elastic_indexer(text, depth, input_url, cluster_id)
   # Handle other file type scraping
   else:
       # If there is no such folder, the script will create one automatically
       folder_location = r"D:\Test folder (2)"
       if not os.path.exists(folder_location):
           os.mkdir(folder_location)
       response = requests.get(input_url)
       soup = BeautifulSoup(response.text, "html.parser")
       test = ''
       i = 0
       extension = '.' + type
       for link in soup.find_all('a',href=True):
           if(link['href'].endswith(extension)):
               print(link['href'])
               filename = os.path.join(
                   folder_location, link['href'].split('/')[-1])
               l = urljoin(input_url, link['href'])
               with open(filename, 'wb') as f:
                   f.write(requests.get(
                       urljoin(input_url, link['href'])).content)
```

```
                read = textract.process(filename, encoding='unicode_escape')
                text = read.decode('utf-8')
                os.remove(filename)
                text = CONTROL_CHAR_RE.sub('', text)
                print("goto indexing")
                elastic_indexer(text, depth, l, cluster_id)
                print("indexed")

    for anchor in beautiful_soup_object.findAll("a"):
        href = anchor.attrs.get("href")
        if(href != "" or href != None):
            href = urljoin(input_url, href)
            href_parsed = urlparse(href)
            href = href_parsed.scheme
            href += "://"
            href += href_parsed.netloc
            href += href_parsed.path
            final_parsed_href = urlparse(href)
            is_valid = bool(final_parsed_href.scheme) and bool(
                final_parsed_href.netloc)
            if is_valid:
                if current_url_domain not in href and href not in links_extern:
                    links_extern.add(href)
                if current_url_domain in href and href not in links_intern:
                    links_intern.add(href)
                    temp_urls.add(href)
    return temp_urls

def crawl(input_url, depth, type, cluster_id):
    try:
        if(depth == 1):
            print("level crawl")
            level_crawler(input_url, depth, type, cluster_id)
            print("scraped!")
        else:
            queue = []
            queue.append(input_url)
            for j in range(depth):
                for count in range(len(queue)):
                    url = queue.pop(0)
```

```
                    urls = level_crawler(url, j, type, cluster_id)
                    for i in urls:
                        queue.append(i)
        return True
    except Exception as e:
        print('Caught exception while crawling ' + str(input_url))
        return False

def elastic_indexer(text, depth, url, cluster_id):
    es_client = Elasticsearch(['http://127.0.0.1:9200'])
    doc = {
        "date": time.strftime("%Y-%m-%d"),
        "currnet_url": url,
        "depth": depth,
        "content": text
    }
    res = es_client.index(index= str(cluster_id), doc_type="_doc", body=doc)
    print("elastic search?:", res["result"])

Python Code File: base_app/sendMail.py
import smtplib
import os
def send(mail):
    gmail_user = os.environ.get('email')
    gmail_password = os.environ.get('mail_pass')
    sent_from = gmail_user
    to = [mail]
    subject = 'Cluster ready'
    body = 'your recent added url is ready for search'
    email_text = """\
    From: %s
    To: %s
    Subject: %s
    %s
    """ % (sent_from, ", ".join(to), subject, body)
    try:
        smtp_server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
        smtp_server.ehlo()
        smtp_server.login(gmail_user, gmail_password)
        smtp_server.sendmail(sent_from, to, email_text)
```

```
        smtp_server.close()
        print("Email sent successfully!")
    except Exception as ex:
        print("Something went wrong….", ex)

Python Code File: base_app/tests.py
 from django.test import TestCase
 # Create your tests here.
 try:
     import elasticsearch
 except ImportError as e:
     print('elasticsearch is not installed')

Python Code File: base_app/updater.py
 from datetime import datetime, timezone
 from apscheduler.schedulers.background import BackgroundScheduler
 from .schedule import scheduled_crawl

 def start():
    # automated and scheduled crawling
    scheduler = BackgroundScheduler(timezone='UTC')
    # setting default time interval = 5 mins
    scheduler.add_job(scheduled_crawl, 'interval', minutes=10)
    scheduler.start()

Python Code File: base_app/urls.py
 from django.urls import path
 from .views import (
    ClusterList, ClusterDetail, ClusterCreate, ClusterUpdate,
    ClusterDelete, UrlList, UrlCreate, SearchView, search
 )
 urlpatterns = [
    path('', ClusterList.as_view(), name='clusters'),
    path('cluster-create/', ClusterCreate.as_view(), name='cluster-create'),
    path('<int:pk>/<str:slug>/', ClusterDetail.as_view(), name='cluster'),
    path('cluster-update/<int:pk>/<str:slug>/', ClusterUpdate.as_view(),
name='cluster-update'),
    path('cluster-delete/<int:pk>/<str:slug>/', ClusterDelete.as_view(),
name='cluster-delete'),
```

```
    path('cluster-search/<int:pk>/<str:slug>/', search, name='cluster-search'),
    path('url-list/', UrlList.as_view(), name='url-list'),
    path('cluster-url/<int:pk>/', UrlCreate.as_view(), name='add-url'),
 ]
```

Python Code File: base_app/views.py

```
 from typing import List
 from django.contrib.auth import models
 from django.core import paginator
 from django.shortcuts import render
 from django.views.generic.list import ListView
 from django.views.generic.detail import DetailView
 from django.views.generic.edit import CreateView, DeleteView, UpdateView
 from django_filters.views import FilterView
 from django.urls import reverse_lazy
 from .forms import UrlForm
 from .models import Cluster, Url
 from django.core.paginator import EmptyPage, PageNotAnInteger, Paginator

 # To create page restrictions
 from django.contrib.auth.mixins import LoginRequiredMixin
 # Elastic Search
 from elasticsearch import Elasticsearch
 from elasticsearch_dsl import Search
 from base_app.es_paginator import SearchResults
 from base_app.export import link
 import requests
 from bs4 import BeautifulSoup

 # A list view of clusters
 class ClusterList(LoginRequiredMixin, FilterView):
    model = Cluster
    context_object_name = 'clusters'
    template_name = 'base_app/clusterList.html'
    paginate_by = 5

    # Restricts user to view other users data
    def get_queryset(self):
        user = self.request.user
        return Cluster.objects.filter(user=user)
```

```
# A detailed view of Cluster
class ClusterDetail(LoginRequiredMixin, DetailView):
   model = Cluster
   context_object_name = 'cluster'
   template_name = 'base_app/clusterDetail.html'
   query_pk_and_slug = True

# Cluster creation view
class ClusterCreate(LoginRequiredMixin, CreateView):
   model = Cluster
   fields = ['title', 'description']
   # After successful creation of Cluster, redirects user to Cluster view
   success_url = reverse_lazy('clusters')
   def form_valid(self, form):
       form.instance.user = self.request.user
       return super(ClusterCreate, self).form_valid(form)

# Update an existing cluster
class ClusterUpdate(LoginRequiredMixin, UpdateView):
    model = Cluster
    fields = ['title', 'description']
    success_url = reverse_lazy('clusters')

    def form_valid(self, form):
       form.instance.user = self.request.user
       return super(ClusterUpdate, self).form_valid(form)

# Delete an existing cluster
class ClusterDelete(LoginRequiredMixin, DeleteView):
   model = Cluster
   context_object_name = 'cluster'
   success_url = reverse_lazy('clusters')

# A list view of urls in a Cluster
class UrlList(FilterView):
   model = Url
   context_object_name = 'urls'
   template_name = 'base_app/url_list.html'
# URL creation view
class UrlCreate(CreateView):
   model = Url
```

```
    form_class = UrlForm

    # After successful creation of URL, redirects user
    def form_valid(self, form):
        url = form.cleaned_data['cluster_url']
        depth = form.cleaned_data['depth']
        id= form.cleaned_data['cluster']
        # print(url, depth, id )
        # begin_crawl(url, depth, id)
        return super(UrlCreate, self).form_valid(form)

    def get_success_url(self):
        return reverse_lazy(
            'cluster',
            kwargs={'pk': self.object.cluster.id,
                    'slug': self.object.cluster.slug}
        )
    def get_form_kwargs(self, *args, **kwargs):
        kwargs = super().get_form_kwargs(*args, **kwargs)
        kwargs['cluster_id'] = self.kwargs.get('pk')
        return kwargs

# Cluster-Search view
class SearchView(LoginRequiredMixin, DetailView):
    model = Cluster
    context_object_name = 'cluster'
    template_name = 'base_app/cluster_search.html'
    query_pk_and_slug = True
    pagination = 10

def search(request, pk, slug):
    es = Elasticsearch()
    es_client = Elasticsearch(['http://127.0.0.1:9200'])
    print('called search')
    search_text = request.GET.get('qq')
    print('Search text: {}'.format(search_text))
    af = request.GET.get('link')
    print('passed url: {}'.format(af))
    print('cluster id/ ealstic search index: {}'.format(pk))
    s = None
    count = 0
```

```
    status = False
    page_obj = None
    print('going to export')
    try:
        if search_text:
            if af is not None:
                print('going to export')
                link(af)
            s = Search(index=str(pk)).using(es_client).query("match",
content=search_text)
            response = s.execute()
            count = 0
            for hit in s:
                count += 1
                print(str(count) + ' - ' + str(hit.currnet_url))
            else:
                print('nothing more found')
                #paginate_by = 3
                search_results = SearchResults(s)
                print(search_results)
        else:
            print('No text to search')
    except Exception as e:
        count = 0
        status = True
        print('slug: {}'.format(slug))
        print('Exception caught')
        msg = 'Warning! Cluster is not yet ready for searching'
        return render(request, 'base_app/cluster_search.html',
                {'cluster':slug, 'warning':msg, 'count':count, 'status':status})

    return render(request, 'base_app/cluster_search.html',
            {'cluster':slug, 'hits':s, 'count':count, 'page_obj':page_obj,
"search": search_text})
 # payment process
 def testpayment(request):
     return render(request, 'donation/testpayment.html')

Python Code File: api/apps.py
 from django.apps import AppConfig
```

```
 class ApiConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'API'

Python Code File: api/serializers.py
 from base_app.models import Cluster, Url
 from rest_framework import serializers
 from rest_framework.permissions import AllowAny
 from django.contrib.auth.models import User
 from rest_framework import serializers
 from rest_framework.permissions import IsAuthenticated
 from rest_framework.serializers import (
    ModelSerializer,
    ValidationError,
    EmailField,
 )
 # Serializer classes for the models
 class cluster_serializer(serializers.ModelSerializer):
    permission_classes = [AllowAny]
    class Meta:
        model = Cluster
        fields = ('user', 'title', 'description', 'date_updated', 'slug')

 class url_serializer(serializers.ModelSerializer):
    permission_classes = [AllowAny]
    class Meta:
        model = Url
        fields = ('cluster','cluster_url','depth','output_type','date_added'
,'slug')
 # Serializer for authentication
 class RegisterSerializer(ModelSerializer):
    email = EmailField(label='Email adress')
    class Meta:
        model = User
        fields = [
            'id','username',
            'password','email',
        ]
    extra_kwargs = {"password": {"write_only":True},
```

```
                    "id":{"read_only":True}
                   }

    def validate(self, data):
        return data
    def validate_email(self, value):
        email = value
        user_qs = User.objects.filter(email=email)
        if user_qs.exists():
            raise ValidationError("Email alredy registred")
        return value

    def create(self, validated_data):
        username = validated_data['username']
        password = validated_data['password']
        email = validated_data['email']
        user_obj = User(
            username = username,
            email = email,
        )
        user_obj.set_password(password)
        user_obj.save()
        return user_obj

Python Code File: api/urls.py
 from django.urls import path
 from django.urls import include, path
 from rest_framework import routers, views
 from .views import ClusterAPI, URL_API, Register
 from django.conf.urls import url
 from rest_framework.authtoken import views as authviews

 urlpatterns =[
    path('clusterapi/', ClusterAPI.as_view()),
    path('urlapi/', URL_API.as_view()),
    path('registerapi/', Register.as_view()),
    # Api authentication endpoint
    path('api-token-auth/', authviews.obtain_auth_token, name='api-token-auth'),
 ]
 from re import I
 import requests
 from django.shortcuts import render
```

```python
from .serializers import cluster_serializer, url_serializer
from django.contrib.auth.models import User

from rest_framework.views import APIView
from rest_framework import viewsets
from rest_framework import permissions
from django.shortcuts import get_object_or_404
from rest_framework import status
from rest_framework.decorators import permission_classes
from rest_framework.response import Response
from base_app.models import Cluster, Url
from django.http import HttpResponse
from django.http import JsonResponse
from .serializers import RegisterSerializer
from rest_framework.generics import
    (CreateAPIView, RetrieveUpdateAPIView, UpdateAPIView,
    DestroyAPIView, ListAPIView, RetrieveAPIView)
from rest_framework.authentication import TokenAuthentication
from rest_framework.authentication import SessionAuthentication
from rest_framework.permissions import IsAuthenticated
from rest_framework.permissions import (
    AllowAny,
   )

# Create your views here.
# Api Views for all the models
class ClusterAPI(APIView):
   permission_classes = [permissions.AllowAny]

   def get(self, request, format=None):
       user=self.request.user
       queryset = Cluster.objects.filter(user=user)
       serializer = cluster_serializer(queryset, many=True)
       return Response(serializer.data)

   def post(self, request, format=None):
       serializer =cluster_serializer(data=request.data)
       if serializer.is_valid():
           serializer.save()
           return Response({'msg': 'Data Created'},
status=status.HTTP_201_CREATED)
       return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

```
 class URL_API(APIView):
    permission_classes = [permissions.AllowAny]
    def get(self, request, format=None):
        queryset = Url.objects.all()
        serializer = url_serializer(queryset, many=True)
        return Response(serializer.data)

    def post(self, request, format=None):
        serializer =url_serializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response({'msg': 'Data Created'},
status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
 # User API view containing all the userinfo for future reference
 class Register(CreateAPIView):
    permission_classes = (AllowAny,)

    serializer_class = RegisterSerializer
    queryset = User.objects.all()


Python Code File: manage.py
 """Django's command-line utility for administrative tasks."""
 import os
 import sys
 def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'googleauth.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

```
 if __name__ == '__main__':
    main()

                                    TEMPLATE
HTML Code File: templates/account/account_inactive.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Account Inactive" %}{% endblock %}
 {% block content %}
     <h1>{% trans "Account Inactive" %}</h1>
     <p>{% trans "This account is inactive." %}</p>
 {% endblock %}

HTML Code File: templates/ account/base.html
 <!DOCTYPE html>
 <html>
   <head>
     <title>{% block head_title %}Re-search{% endblock %}</title>
     {% block extra_head %}
     <!-- Compiled and minified JavaScript -->
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
 </script>

<scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materiali
ze.min.js">
 </script>
    <script>
      $(document).ready(function(){
        $('select').formSelect();
      });
    </script>
    <!--Import Google Icon Font-->
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
    <!-- Compiled and minified CSS -->
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.mi
n.css">
```

```
    <!-- Compiled and minified JavaScript -->
    <scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/mater
ialize.min.js"></script>
    <link href='https://fonts.googleapis.com/css?family=Merriweather Sans'
rel='stylesheet'>
    {% endblock %}
  </head>
  <body>
    {% block body %}
    <div>
      <nav>
        <div class="nav-wrapper" style="background-color: dodgerblue;">
          <a href="/cluster" class="brand-logo" style="font-weight: bolder;">
  RE-SEARCH</a>
          <ul id="nav-mobile" class="right hide-on-med-and-down">
            <ul>
              {% if user.is_authenticated %}
              <li><i class="material-icons">account_circle</i></li>
              <li> {{user}}</li>
              <li><a href="{% url 'account_email' %}">Change E-mail</a></li>
              <li><a href="{% url 'account_logout' %}">Sign Out</a></li>
              <li><a href= "{% url 'payment' %}" >Donate</a></li>
              {% else %}
              <li><a href="{% url 'account_login' %}">Sign In</a></li>
              <li><a href="{% url 'account_signup' %}">Sign Up</a></li>
              <li><a href= "{% url 'payment' %}" >Donate</a></li>
              {% endif %}
            </ul>
          </ul>
        </div>
      </nav>
      {% if messages %}
        <div>
          <strong>Messages:</strong>
          {% for message in messages %}
              <li>{{message}}</li>
          {% endfor %}
          </div>
      {% endif %}
    </div>
```

```
    <div class="container">
      {% block content %}
      {% endblock %}
    </div>
     <!-- Include the compiled and minified Materialize JavaScript -->
        <scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.97.5/js/
materialize.min.js">
        </script>
    {% endblock %}
    {% block extra_body %}
    {% endblock %}
  </body>
 </html>

HTML Code File: templates/ account/email_confirm.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% load account %}
 {% block head_title %}{% trans "Confirm E-mail Address" %}{% endblock %}
 {% block content %}
     <h1>{% trans "Confirm E-mail Address" %}</h1>
 {% if confirmation %}
 {% user_display confirmation.email_address.user as user_display %}
 <p>{% blocktrans with confirmation.email_address.email as email %}Please confirm
that <a href="mailto:{{ email }}">{{ email }}</a> is an e-mail address for user
{{ user_display }}.{% endblocktrans %}</p>

 <form method="post" action="{% url 'account_confirm_email' confirmation.key %}">
 {% csrf_token %}
    <button class="btn btn-primary" type="submit">{% trans 'Confirm' %}</button>
 </form>
 {% else %}
 {% url 'account_email' as email_url %}

<p>{% blocktrans %}This e-mail confirmation link expired or is invalid. Please
<a href="{{ email_url }}">issue a new e-mail confirmation request</a>.{%
endblocktrans %}</p>
 {% endif %}
 {% endblock %}
```

```
HTML Code File: templates/account/email.html
 {% extends "account/base.html" %}
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% block head_title %}{% trans "E-mail Addresses" %}{% endblock %}
 {% block content %}
 <style>
  .btn-primary{
    background-color: dodgerblue;
  }
 </style>
    <h4 style="font-weight: bold;">{% trans "E-mail Addresses" %}</h4>
 {% if user.emailaddress_set.all %}<p> {% trans 'The following e-mail addresses
are associated with your account:'%}</p>
 <form action="{% url 'account_email' %}" class="email_list" method="post">
 {% csrf_token %}
 <fieldset class="blockLabels">
  {% for emailaddress in user.emailaddress_set.all %}
 <div class="ctrlHolder">
      <label style="font-size: larger;" for="email_radio_{{forloop.counter}}"
class="{% if emailaddress.primary %}primary_email{%endif%}">

      <input class="with-gap" id="email_radio_{{forloop.counter}}" type="radio"
name="email" {% if emailaddress.primary or user.emailaddress_set.count == 1
%}checked="checked"{%endif %} value="{{emailaddress.email}}"/>

 {{emailaddress.email}}
    {% if emailaddress.verified %}
    <span class="verified">{% trans "Verified" %}</span>
    {% else %}
    <span class="unverified">{% trans "Unverified" %}</span>
    {% endif %}
 {% if emailaddress.primary %}<span class="primary">{% trans "Primary" %}</span>
 {% endif %}
 </label>
 </div>
  {% endfor %}
 <div class="buttonHolder">
      <button class="btn btn-primary" class="secondaryAction" type="submit"
name="action_primary" >{% trans 'Make Primary' %}</button>
```

```
      <button class="btn btn-primary" class="secondaryAction" type="submit"
name="action_send" >{% trans 'Re-send Verification' %}</button>
      <button class="btn btn-primary" class="primaryAction" type="submit"
  style="background-color:red" name="action_remove" >{% trans 'Remove'%}</button>
  </div>
  </fieldset>
</form>
 {% else %}
<p><strong>{% trans 'Warning:'%}</strong> {% trans "You currently do not have any
e-mail address set up. You should really add an e-mail address so you can receive
notifications, reset your password, etc." %}</p>
 {% endif %}
  {% if can_add_email %}
    <h4 style="font-weight: bold;">{% trans "Add E-mail Address" %}</h4>
    <form method="post" action="{% url 'account_email' %}" class="add_email">
        {% csrf_token %}
        {{form|crispy}} <button name="action_add" type="submit" class="btn btn-
primary">{% trans "Add E-mail" %}</button>
    </form>
  {% endif %}
 {% endblock %}
 {% block extra_body %}
 <script type="text/javascript">
 (function() {
  var message = "{% trans 'Do you really want to remove the selected e-mail
address?' %}";
  var actions = document.getElementsByName('action_remove');
  if (actions.length) {
    actions[0].addEventListener("click", function(e) {
      if (! confirm(message)) {
        e.preventDefault();
      }
    });
  }
 })();
 </script>
 {% endblock %}

HTML Code File: templates/account/login.html
 {% extends "account/base.html" %}
```

```
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% load account socialaccount %}
 {% block head_title %}{% trans "Sign In" %}{% endblock %}
 {% block content %}
 <style>
  #container{
      text-align: center;
  }
 </style>
 <div id="container" class="z-depth-2" style="width: max-content; border-radius:
7px; position: absolute; top: 25%; left: 33%; width: 500px; margin-top: -50px;
margin-left: -50px; background-color:white">
    <h3 style="font-family: Merriweather Sans; font-weight: bold;">
    {% trans "Sign In" %}</h3>
 {% get_providers as socialaccount_providers %}

 {% if socialaccount_providers %}
 <p>{% blocktrans with site.name as site_name %}Please sign in with one
 of your existing third party accounts. <br>Or, <a href="{{ signup_url }}">Sign
up</a> for a {{ site_name }} account and sign in below:{% endblocktrans %}</p>

 <div class="socialaccount_ballot">
  <ul class="socialaccount_providers">
    <img width="30px" style="margin-bottom:3px; margin-right:5px" alt="Google
sign-in"
src="https://upload.wikimedia.org/wikipedia/commons/thumb/5/53/Google_%22G%22_Log
o.svg/512px-Google_%22G%22_Logo.svg.png" />
  {% include "socialaccount/snippets/provider_list.html" with process="login" %}
  </ul> <div class="login-or"></div>
 </div>
 {% include "socialaccount/snippets/login_extra.html" %}
 {% else %}

 <p>{% blocktrans %}If you have not created an account yet, then please
 <a href="{{ signup_url }}">sign up</a> first.{% endblocktrans %}</p>
 {% endif %}

 <form class="login" method="POST" action="{% url 'account_login' %}"
style="margin: 50px; margin-top:20px">
```

```
  {% csrf_token %}
  {{ form|crispy }}
  {% if redirect_field_value %}
  <input type="hidden" name="{{ redirect_field_name }}" value="{{
redirect_field_value }}" />
  {% endif %}
  <a class="button secondaryAction" href="{% url 'account_reset_password' %}">
    {% trans "Forgot Password?" %}</a>  
  <button class="btn btn-primary" type="submit"
   style="background-color:  dodgerblue;">
    {% trans "Sign In" %}</button>
 </form>
 {% endblock %}

HTML Code File: templates/account/logout.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Sign Out" %}{% endblock %}
 {% block content %}
 <h1 style="font-family: Merriweather Sans; font-weight: bold;">
 {% trans "Sign Out" %}</h1>
 <p style="font-size: larger;">
 {% trans 'Are you sure you want to sign out?' %}</p>

 <form method="post" action="{% url 'account_logout' %}">
  {% csrf_token %}
  {% if redirect_field_value %}
  <input type="hidden" name="{{ redirect_field_name }}"
  value="{{ redirect_field_value }}"/>
  {% endif %}
  <button type="submit" class="btn btn-primary"
   style="background-color: dodgerblue;">
    {% trans 'Sign Out' %}</button>
 </form>
 {% endblock %}

 HTML Code File: templates/account/password_change.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Change Password" %}{% endblock %}
```

```
 {% block content %}
    <h1>{% trans "Change Password" %}</h1>
    <form method="POST" action="{% url 'account_change_password' %}"
class="password_change">
        {% csrf_token %}
        {{form.as_p}}
 <button type="submit" name="action">{% trans "Change Password" %}</button>
 <a href="{% url 'account_reset_password' %}">{% trans "Forgot Password?" %}</a>
 </form>
 {% endblock %}

HTML Code File: templates/account/password_reset_done.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% load account %}
 {% block head_title %}{% trans "Password Reset" %}{% endblock %}
 {% block content %}
    <h1>{% trans "Password Reset" %}</h1>
    {% if user.is_authenticated %}
    {% include "account/snippets/already_logged_in.html" %}
    {% endif %}
    <p>{% blocktrans %}We have sent you an e-mail.
    Please contact us if you do  not receive it within a few minutes.
 {% endblocktrans %}</p>
 {% endblock %}

HTML Code File: templates/account/password_reset_from_key_done.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Change Password" %}{% endblock %}
 {% block content %}
    <h1>{% trans "Change Password" %}</h1>
    <p>{% trans 'Your password is now changed.' %}</p>
 {% endblock %}

HTML Code File: templates/account/password_reset_from_key.html
 {% extends "account/base.html" %}
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% block head_title %}{% trans "Change Password" %}{% endblock %}
```

```
 {% block content %}
    <h1>{% if token_fail %}{% trans "Bad Token" %}{% else %}
    {% trans "Change Password" %}{% endif %}</h1>
    <div>
    {% if token_fail %}
        {% url 'account_reset_password' as passwd_reset_url %}
        <p>{% blocktrans %}The password reset link was invalid, possibly
         because it has already been used.  Please request a
 <a href="{{ passwd_reset_url }}">new password reset</a>.{% endblocktrans %}</p>
    {% else %}
        {% if form %}
            <form method="POST" action="{{ action_url }}">
                {% csrf_token %}
                {{ form|crispy }}
 <input class="btn btn-primary" type="submit" name="action" value="{% trans'
Change password' %}"/>
            </form>
        {% else %}
            <p>{% trans 'Your password is now changed.' %}</p>
        {% endif %}
    {% endif %}
  </div>
 {% endblock %}

HTML Code File: templates/account/password_reset.html
 {% extends "account/base.html" %}
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% load account %}

 {% block head_title %}{% trans "Password Reset" %}{% endblock %}
 {% block content %}

    <h1>{% trans "Password Reset" %}</h1>
    {% if user.is_authenticated %}
    {% include "account/snippets/already_logged_in.html" %}
    {% endif %}
    <p>{% trans "Forgotten your password? Enter your e-mail address below,
    and we'll send you an e-mail allowing you to reset it." %}</p>
    <form method="POST" action="{% url 'account_reset_password' %}"
```

```
 class="password_reset">
        {% csrf_token %}
        {{ form|crispy }}
    <input type="submit" value="{% trans 'Reset My Password' %}"
    class="btn btn-primary"/>
 </form>
    <p>{% blocktrans %}Please contact us if you have any trouble resetting your
password.{% endblocktrans %}</p>
 {% endblock %}

HTML Code File: templates/account/password_set.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Set Password" %}{% endblock %}
 {% block content %}
    <h1>{% trans "Set Password" %}</h1>
    <form method="POST" action="{% url 'account_set_password' %}"
class="password_set">
        {% csrf_token %}
        {{ form.as_p }}
        <input type="submit" name="action" value="{% trans 'Set Password' %}"/>
    </form>
 {% endblock %}

HTML Code File: templates/account/signup_closed.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Sign Up Closed" %}{% endblock %}
 {% block content %}
     <h1>{% trans "Sign Up Closed" %}</h1>
     <p>{% trans "We are sorry, but the sign up is currently closed." %}</p>
 {% endblock %}

HTML Code File: templates/account/signup.html
 {% extends "account/base.html" %}
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% block head_title %}{% trans "Signup" %}{% endblock %}
 {% block content %}
```

```
 <style>
  #container{
      text-align: center;
  }
 </style>
 <div id="container" class="z-depth-2" style="width: max-content;
 border-radius: 7px; position: absolute; top: 20%; left: 33%; width: 500px;
 margin-top: -50px; margin-left: -50px; background-color:white">
 <h3 style="font-family: Merriweather Sans; font-weight: bold;">
 {% trans "Sign Up" %}</h3>

 <p>{% blocktrans %}Already have an account? Then please
 <a href="{{ login_url }}">sign in</a>.{% endblocktrans %}</p>
 <form class="signup" id="signup_form" method="post" action="{% url
'account_signup' %}" style="margin: 50px; margin-top:20px">
  {% csrf_token %}
  {{ form|crispy }}
  {% if redirect_field_value %}
  <input type="hidden" name="{{ redirect_field_name }}" value="{{
redirect_field_value }}" />
  {% endif %}
  <button type="submit" class="btn btn-primary" style="background-color:
dodgerblue;">
    {% trans "Sign Up" %} &raquo;</button>
   </form>
  </div>
 {% endblock %}

HTML Code File: templates/account/verification_send.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Verify Your E-mail Address" %}{% endblock %}
 {% block content %}
    <h1>{% trans "Verify Your E-mail Address" %}</h1>
    <p>{% blocktrans %}We have sent an e-mail to you for verification.
    Follow the link provided to finalize the signup process. Please contact us
    if you do not receive it within a few minutes.{% endblocktrans %}</p>
 {% endblock %}

HTML Code File: templates/account/verified_email_required.html
```

```
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Verify Your E-mail Address" %}{% endblock %}
 {% block content %}
     <h1>{% trans "Verify Your E-mail Address" %}</h1>
 {% url 'account_email' as email_url %}
 <p>{% blocktrans %}This part of the site requires us to verify that
 you are who you claim to be. For this purpose, we require that you
 verify ownership of your e-mail address. {% endblocktrans %}</p>

 <p>{% blocktrans %}We have sent an e-mail to you for
 verification. Please click on the link inside this e-mail. Please
 contact us if you do not receive it within a few minutes.{% endblocktrans %}</p>

    <p>{% blocktrans %}<strong>Note:</strong> you can still
    <a href="{{ email_url  }}">change your e-mail address</a>.
   {% endblocktrans %}</p>
 {% endblock %}

HTML Code File: templates/base/index.html
 {% extends "account/base.html" %}
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% block content %}
    <div>
<h3 style="font-family: Merriweather Sans; font-weight: bold; color:dodgerblue;">
  {% trans "RE-SEARCH" %}</h3>
        {% if user.is_authenticated %}
        <p><h4 style="font-family: Merriweather Sans; font-weight: bold;">
        Hello, {{user.username}}<br>Welcome to Re-Search</h4></p><br>
        <p><a href="cluster/"><button class="waves-effect waves-light btn-large"
        style="background-color: dodgerblue;">View My Clusters</button></a><p>
        {% else %}
        <p><h3 style="font-family: Merriweather Sans; font-weight: bold;">
        Welcome to Re-Search</h3><br><a href="accounts/login/">
        <button class="waves-effect waves-light btn-large"
        style="background-color: dodgerblue;">
                Sign in</button></a> to continue.</p>
        {% endif %}
    </div>
 {% endblock %}
```

```
HTML Code File: templates/base_app/cluster_confirm_delete.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% block head_title %}{% trans "Delete Cluster" %}{% endblock %}
 {% block content %}
 <br><br>
    <a href="{% url 'clusters' %}" class="btn-floating btn-tiny
    waves-effect waves-light blue">
 <i class="material-icons">arrow_back</i></a>
 <h2 style="font-weight: bold;">{% trans "Warning" %}</h2>

 <form method="POST">
    {% csrf_token %}
    {% if redirect_field_value %}
    <input type="hidden" name="{{ redirect_field_name }}"
    value="{{ redirect_field_value }}"/>
    {% endif %}
    <p>Once you delete the cluster all the data related to this cluster
    will be lost.<br> Do you really want to delete this Search Cluster?</p>
    <input type="submit" value="Delete" class="btn btn-primary"
style="background-color: red;"/>
  </form>
 {% endblock %}

HTML Code File: templates/base_app/cluster_form.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% load crispy_forms_tags %}
 {% block head_title %}{% trans "Create or Update Cluster" %}{% endblock %}
 {% block content %}
 <style>
    #container{
        text-align: center;
    }
 </style>
 <br><br><br><br>   
 <a href="{% url 'clusters' %}" class="btn-floating btn-tiny
 waves-effect waves-light blue">
    <i class="material-icons">arrow_back</i></a>
```

```
 <div id="container" class="z-depth-2" style="width: max-content;
 border-radius: 7px;position: absolute; top: 30%; left: 25%; width: 800px;
 margin-top: -50px; margin-left: -50px; background-color:white">
 <h3 style="font-family: Merriweather Sans; font-weight: bold;">
 {% trans "Create  or update a Search Cluster" %}</h3>

 <form method="POST" action="" style="margin: 50px;">
    {% csrf_token %}
    {{ form|crispy }}
    {% if redirect_field_value %}
           <input type="hidden" name="{{ redirect_field_name }}"
           value="{{ redirect_field_value }}"/>
    {% endif %} <br>
    <input type="submit" value="Submit" class="btn btn-primary"
style="background-color: dodgerblue;"/>
    </form>
 </div>

{% endblock %}

HTML Code File: templates/base_app/cluster_search.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% load crispy_forms_tags %}
 {% block head_title %}{% trans "Search" %}{% endblock %}
 {% block content %} <br>
     <a href="{% url 'clusters' %}" class="btn-floating btn-tiny
     waves-effect waves-light blue">
     <i class="material-icons">arrow_back</i></a>
 <P style="font-size:17px; font-family: Merriweather Sans;">
     You are searching in {{cluster.title}}</P>
 <form method="GET" action=""  >
    {% csrf_token %}
    <button class="btn btn-primary" style="background-color: dodgerblue;">
Search</button>
    {% if search is None %}
        <input id="q" name="qq" type="text"   placeholder="Enter search
        text here" style="width: 500px;">
    {%else%}
        <input id="q" name="qq" type="text" placeholder="Enter search text here"
value="{{search}}"  style="width: 500px;">
```

```
    {%endif%}
    <input type="hidden" id='link' name="link" value="">
   <br>
  </form>
 <form method="GET" action="">
    <button class="btn-floating btn-tiny waves-effect
    waves-light blue" action="">
        <i class="material-icons">keyboard_voice</i></button>
        <!-- Input area for voice search-->
        <input name="qq" type="text" name="" id="speechToText"
      placeholder="Speak Something" onclick="record()" style="width: 545px;">
 </form>
    <!-- Below is the script for voice recognition and conversion to text-->
    <script>
        function record() {
            var recognition = new webkitSpeechRecognition();
            recognition.lang = "en-GB";
            recognition.onresult = function(event) {
                document.getElementById('speechToText').value =
event.results[0][0].transcript;
            }
          recognition.start();
        }
     </script>
 <script>
    function getURL() {
        let a = window.location.href
        return a
    }
    document.getElementById("link").value = getURL()
 </script>
 {% if hits %}
    {% for hit in hits%}
        {%if not empty and forloop.counter <= 1 %}
            <a href="D:\Test folder (2)\demo1.csv" download>
            <button class="btn btn-primary" style="margin-left: 1000px;">
            Export as CSV</button></a>
        {%endif%}
    {% endfor %}
 {% endif %}
```

```
 <div style="margin: 5px;">
    {% if hits %}
        <table>
            {% for hit in hits %}<tr>
                <td class='lin'>Hit: <a href="{{hit.currnet_url}}"
target="_blank">{{hit.currnet_url}}</a></td>
            </tr> <tr>
                <td class='content'><p>Content: {{hit.content}}</p></td>
            </tr>
            {% empty %}
                <h3> No search result found </h3>
            {% endfor %}
        </table>

    {% endif %}
 </div>
 {% if status %}
    <h5 style="color: red; font-family: Merriweather Sans;">{{warning}}</h5>
    <P style="font-size:18px; font-family: Merriweather Sans;" >
    Possible reasons are:
        <ul style="list-style-type:disc; font-size:18px;">
            <li> The Cluster doesn't have any url</li>
            <li> Existing url(s) are not yet crawled</li>
        </ul>
    </P>
 {% endif %}
 {% endblock %}

HTML Code File: templates/base_app/clusterDetail.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% load crispy_forms_tags %}
 {% block head_title %}{% trans "Cluster Details" %}{% endblock %}
 {% block content %}
 <style>
    #container{
        text-align: center;
    }
 </style>
```

```
 <br><br>
 <a href="{% url 'clusters' %}" class="btn-floating btn-tiny waves-effect waves-
light blue"> <i class="material-icons">arrow_back</i></a>

 <div id="container" class="z-depth-2" style="width: max-content; border-radius:
7px; position: absolute; top: 20%; left: 25%; width: 850px;
    margin-top: -50px; margin-left: -50px; background-color:white">

    <h3 style="font-weight: bold;">Cluster: {{cluster}}</h3>
    <p>Created on: {{cluster.date_created}} in GMT, UTC +0</p>
    <p>Last update: {{cluster.date_updated}} in GMT, UTC +0</p>
    <p>Description: {{cluster.description}}</p>
    <a href="{% url 'add-url' pk=cluster.id%}">
    <button class="btn btn-primary" style="background-color: dodgerblue;">
    Add URL</button></a>
    <h4 style="font-weight: bold;">{% trans "List of added url: " %}</h4>

    {% for url in cluster.urls.all %}
            URL : <a href="{{url.cluster_url}}">{{url.cluster_url}}</a><br>
            Crawl Method: {{url.output_type}}    
            Depth: {{url.depth}}     Date Added: {{url.date_added}}<br>
            <br>
    {% empty %}
        <p>You did not add any url</p>
    {% endfor %}
    <br><br>
 </div>
 {% endblock %}

HTML Code File: templates/base_app/clusterList.html
 {% extends "account/base.html" %}
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% block head_title %}{% trans "Clusters" %}{% endblock %}
 {% block content %}
 <style>
    #container{
        text-align: center;
    }
 </style>
```

```
 <div id="container" class="z-depth-2" style="width: max-content; border-radius:
7px; position: absolute; top: 20%; left: 28%; width: 700px;
    margin-top: -50px; margin-left: -50px; background-color:white">

 <h3 style="font-family: Merriweather Sans; text-align:center; font-weight:
bold;">{% trans "Search Clusters" %}</h3>
 <a href="{% url 'cluster-create' %}">
 <button class="btn btn-primary" style="background-color: dodgerblue;">
       Create Search Cluster</button></a>
 <p style="font-size:17px; text-align:center">Oredered by: Date Created</p>
    <table class="highlight" style="width: max-content;" align="center">
        {% for cluster in clusters %}
        <tr>
            <td style="font-size:17px; ">{{cluster.title}}   </td>
            <td><a href="{% url 'cluster-search'
            pk=cluster.id slug=cluster.slug %}">
            <button class="btn btn-primary"
            style="background-color: dodgerblue;"> Search</button></a>

     <a href="{% url 'cluster' pk=cluster.id slug=cluster.slug %}">
     <button class="btn btn-primary">View</button></a>
     <a href="{% url 'cluster-update' pk=cluster.id slug=cluster.slug %}">
     <button class="btn btn-primary" style="background-color:blueviolet;">
                        Edit</button></a>
     <a href="{% url 'cluster-delete' pk=cluster.id slug=cluster.slug %}">
     <button class="btn btn-primary" style="background-color:red;">
                        Delete</button></a>
       </td>
     </tr>
        {% empty %}
            <h3> You do not have any cluster </h3>
        {% endfor %}
    </table>
    <br><br>
    <div class="pagination">
        <span class="step-links">
            {% if page_obj.has_previous %}
    <a href="?page=1" class="btn-floating btn-tiny waves-effect waves-light red">
                    <i class="material-icons">arrow_back</i></a>
```

```
    <a href="?page={{ page_obj.previous_page_number }}" class="btn-floating btn-
tiny  waves-effect waves-light blue">
   <i class="material-icons">navigate_before</i></a>
 {% endif %}
 <span class="current"> Page {{ page_obj.number }} of {{
page_obj.paginator.num_pages }}.</span>

  {% if page_obj.has_next %}
     <a href="?page={{ page_obj.next_page_number }}"
     class="btn-floating btn-tiny waves-effect waves-light blue">
     <i class="material-icons">navigate_next</i></a>

     <a href="?page={{ page_obj.paginator.num_pages }}"
     class="btn-floating btn-tiny waves-effect waves-light red">
                    <i class="material-icons">arrow_forward</i></a>
           {% endif %}
        </span>
      </div>
    <br>
  </div>
 {% endblock %}

HTML Code File: templates/base_app/url_form.html
 {% extends "account/base.html" %}
 {% load i18n %}
 {% load crispy_forms_tags %}
 {% block head_title %}{% trans "Add a new url" %}{% endblock %}
 {% block content %}
 <style>
    #container{
        text-align: center;
    }
 </style>
 <br><br><br>       
 <a href="{% url 'clusters' %}"
 class="btn-floating btn-tiny waves-effect waves-light blue">
    <i class="material-icons">arrow_back</i></a>
 <div id="container" class="z-depth-2"
 style="width: max-content; border-radius: 7px;
    position: absolute; top: 25%; left: 28%; width: 700px;
```

```
    margin-top: -50px; margin-left: -50px; background-color:white">
 <h3 style="font-family: Merriweather Sans; font-weight: bold;">
 {% trans "Add an URL" %}</h3>
 <form method="POST" action="" style="margin: 50px; margin-top:20px">
    {% csrf_token %}
    {{ form|crispy }}
    {% if redirect_field_value %}
    <input type="hidden" name="{{ redirect_field_name }}"
    value="{{ redirect_field_value }}"/>
    {% endif %} <br>
    <input type="submit" value="Submit" class="btn btn-primary"
style="background-color: dodgerblue;"/>
  </form>
 {% endblock %}

HTML Code File: templates/base_app/url_list.html
 {% extends "account/base.html" %}
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% block head_title %}{% trans "Clusters" %}{% endblock %}
 {% block content %}
 <a href="{% url 'url-create' %}">Add url</a>
 <table>
    {% for url in urls %}
     <tr>
        <td>url : {{url.cluster_url}}</td>
     </tr>
    {% empty %}
        <h3> You did not add any url yet </h3>
    {% endfor %}
 </table>
 <!--pagination-->
 <div class="pagination">
    <span class="step-links">
        {% if page_obj.has_previous %}
            <a href="?page=1">&laquo; first</a>
            <a href="?page={{ page_obj.previous_page_number }}">previous</a>
        {% endif %}
        <span class="current">
            Page {{ page_obj.number }} of {{ page_obj.paginator.num_pages }}.
        </span>
```

```
        {% if page_obj.has_next %}
            <a href="?page={{ page_obj.next_page_number }}">next</a>
            <a href="?page={{ page_obj.paginator.num_pages }}">last &raquo;</a>
        {% endif %}
    </span>
  </div>
 {% endblock %}

HTML Code File: templates/donation/testpayment.html
 {% extends "account/base.html" %}
 {% load crispy_forms_tags %}
 {% load i18n %}
 {% block content %}
 <p style="text-align: center;">RE-SEARCH is a FREE search engine which is in
under development process.
 <br>Our goal is to make RE-SEARCH more reliable, efficient and faster.
  <br> Support us by donating - 9.99$....</p>
  <script src="https://www.paypal.com/sdk/js?client-id=test"></script>
        <!-- Set up a container element for the button -->
  <div id="paypal-button-container" style="padding-top: 30px;
   padding-left: 300px; padding-right: 300px; padding-bottom: 300px;"></div>
  <script>
       paypal.Buttons({
           // Sets up the transaction when a payment button is clicked
           createOrder: (data, actions) => {
           return actions.order.create({
               purchase_units: [{
               amount: {
                   value: '9.99' // Can also reference a variable or function
               }
               }]
           });
           },
           // Finalize the transaction after payer approval
           onApprove: (data, actions) => {
           return actions.order.capture().then(function(orderData) {
               // Successful capture! For dev/demo purposes:
 console.log('Capture result', orderData, JSON.stringify(orderData, null, 2));
 const transaction = orderData.purchase_units[0].payments.captures[0];
```

```
alert(`Transaction ${transaction.status}: ${transaction.id}\n\n\
See console for all available details`);
          });
          }
      }).render('#paypal-button-container');
      </script>
{% endblock %}
```