**Appendix**

**view.py**

# My Django imports

from django.shortcuts import render, redirect, reverse

from django.views import View

from django.contrib.auth import authenticate, login, logout

from datetime import date

from django.contrib import messages

from django.contrib.messages.views import SuccessMessageMixin

from django.core.exceptions import ObjectDoesNotExist

from django.views.generic import ListView, DetailView, CreateView, UpdateView, DeleteView

from django.http import HttpResponse

from django.contrib.auth.mixins import LoginRequiredMixin

# My App imports

from e\_verify\_auth.models import Accounts

from e\_verify\_app.models import ResultInformation

from e\_verify\_auth.forms import AccountCreationForm, OrganizationForm, AccountUpdateForm, OrganizationUpdateForm, ResultForm, EditResultForm

# Create your views here.

class DashboardView(LoginRequiredMixin, View):

    login\_url = 'auth:login'

    def get(self, request):

        context = {

            'time':date.today().strftime("%Y-%m-%d"),

            'admin':Accounts.objects.all().count(),

            'clients':Accounts.objects.filter(is\_staff=False).count(),

            'cert': ResultInformation.objects.all().count(),

        }

        return render(request, 'auth/dashboard.html', context)

class LoginView(View):

    def get(self, request):

        return render(request, 'auth/login.html')

    def post(self, request):

        email = request.POST.get('email').strip().lower()

        password = request.POST.get('password').strip()

        if email and password:

            # Authenticate user

            user = authenticate(request, email=email, password=password)

            if user:

                if user.is\_active:

                    login(request, user)

                    messages.success(request, f'You are now signed in {user}')

                    return redirect('auth:dashboard')

                else:

                    messages.warning(request, 'Account not active contact the administrator')

                    return redirect('auth:login')

            else:

                messages.warning(request, 'Invalid login credentials')

                return redirect('auth:login')

        else:

            messages.error(request, 'All fields are required!!')

            return redirect('auth:login')

class CreateAdminView(LoginRequiredMixin, SuccessMessageMixin, CreateView):

    login\_url = 'auth:login'

    model = Accounts

    form\_class = AccountCreationForm

    template\_name = 'auth/create\_admin.html'

    success\_message = "Account created successfully!"

    def get\_success\_url(self):

        return reverse("auth:manage\_admin")

    def form\_valid(self, form):

        form.instance.set\_password(form.instance.password)

        form.instance.email = form.instance.email.strip().lower()

        form.instance.is\_staff = True

        return super().form\_valid(form)

class ManageAdminView(LoginRequiredMixin, ListView):

    login\_url = 'auth:login'

    model = Accounts

    template\_name = "auth/manage\_admin.html"

    def get\_queryset(self):

        return Accounts.objects.filter(is\_staff=True).order\_by('-date\_joined')

class LogoutView(LoginRequiredMixin, View):

    login\_url = 'auth:login'

    def post(self, request):

        logout(request)

        messages.success(request, 'You are successfully logout, to continue login again')

        return redirect('auth:login')

class RegisterView(SuccessMessageMixin, CreateView):

    model = Accounts

    form\_class = OrganizationForm

    template\_name = 'auth/register.html'

    success\_message = "Account created successfully! You can now login!"

    def get\_success\_url(self):

        return reverse("auth:login")

    def form\_valid(self, form):

        form.instance.set\_password(form.instance.password)

        form.instance.email = form.instance.email.strip().lower()

        return super().form\_valid(form)

class CreateOrgView(LoginRequiredMixin, RegisterView):

    login\_url = 'auth:login'

    template\_name = 'auth/create\_org.html'

    def get\_success\_url(self):

        return reverse("auth:manage\_org")

class ManageOrgUserView(ManageAdminView):

    login\_url = 'auth:login'

    template\_name = "auth/manage\_org.html"

    def get\_queryset(self):

        return Accounts.objects.filter(is\_staff=False).order\_by('-date\_joined')

class DeleteUserView(LoginRequiredMixin, SuccessMessageMixin, DeleteView):

    login\_url = 'auth:login'

    model = Accounts

    success\_message = "Account deleted successfully!"

    def get\_success\_url(self):

        return reverse("auth:manage\_admin")

class DeleteOrgView(DeleteUserView):

    login\_url = 'auth:login'

    def get\_success\_url(self):

        return reverse("auth:manage\_org")

class ProfileView(LoginRequiredMixin, SuccessMessageMixin, View):

    login\_url = 'auth:login'

    def get(self, request, pk):

        try:

            user = Accounts.objects.get(pk=pk)

            if user.is\_staff:

                context = {

                    'form': AccountUpdateForm(instance=user),

                    'user': user,

                }

            else:

                context = {

                    'form': OrganizationUpdateForm(instance=user),

                    'user': user,

                }

            return render(request, 'auth/profile.html', context)

        except ObjectDoesNotExist:

            messages.error(request, 'User account not found!')

            return redirect('auth:dashboard')

    def post(self, request, pk):

        try:

            user = Accounts.objects.get(pk=pk)

            if 'password' in request.POST:

                password1 = request.POST.get('password1')

                password2 = request.POST.get('password2')

                if user.is\_staff:

                    context = {

                    'form': AccountUpdateForm(instance=user),

                    'user': user,

                }

                else:

                    context = {

                        'form': OrganizationUpdateForm(instance=user),

                        'user': user,

                    }

                if password1 and password2:

                    if password1 != password2:

                        messages.error(request, 'Passwords does not match!')

                        return redirect('auth:profile', pk)

                    if len(password1) < 6 :

                        messages.error(request, 'Password too short, ensure at least 6 characters!')

                        return redirect('auth:profile', pk)

                    user.set\_password(password1)

                    user.save()

                    messages.success(request, 'Password reset successful!!')

                    if request.user == user:

                        return redirect('auth:login')

                    if request.user.is\_superuser:

                        return redirect('auth:profile', pk)

                    return redirect('auth:login')

            else:

                if user.is\_staff:

                    form = AccountUpdateForm(request.POST, request.FILES, instance=user)

                else:

                    form = OrganizationUpdateForm(request.POST, request.FILES, instance=user)

                if form.is\_valid() :

                    form.save()

                    messages.success(request, 'Profile updated successfully!')

                    return redirect('auth:profile', pk)

                messages.error(request, 'your response contains invalid data!')

                return render(request, 'auth/profile.html', {'form':form, 'user':user})

        except ObjectDoesNotExist:

            messages.error(request, 'User account not found!')

            return redirect('auth:dashboard')

class UploadResultView(LoginRequiredMixin, SuccessMessageMixin, CreateView):

    login\_url = 'auth:login'

    model = ResultInformation

    form\_class = ResultForm

    template\_name = 'auth/upload\_result.html'

    success\_message = "Result Information uploaded!"

    def get\_success\_url(self):

        return reverse("auth:manage\_result")

class ManageResultView(LoginRequiredMixin, View):

    login\_url = 'auth:login'

    def get(self, request):

        return render(request, 'auth/manage\_result.html')

class ListResultView(ManageAdminView):

    login\_url = 'auth"login'

    template\_name = "partials/result\_list.html"

    def get\_queryset(self):

        return ResultInformation.objects.all().order\_by('-date')

class ResultEditForm(LoginRequiredMixin, View):

    login\_url = 'auth:login'

    def get(self, request, pk):

        result = ResultInformation.objects.get(pk=pk)

        form = EditResultForm(instance=result)

        return render(request, 'auth/result\_form.html', {'form':form, 'result':result})

    def post(self, request, pk):

        result = ResultInformation.objects.get(pk=pk)

        form = EditResultForm(request.POST, instance=result)

        if form.is\_valid():

            form.save()

            messages.success(request, 'Result has been updated!')

            return HttpResponse(status=204, headers={'Hx-Trigger':'listChanged'})

        messages.error(request, f'{form.errors.as\_text()}')

        return HttpResponse(status=204, headers={'Hx-Trigger':'listChanged'})

class DeleteResultView(LoginRequiredMixin, SuccessMessageMixin, DeleteView):

    login\_url = 'auth:login'

    model = ResultInformation

    success\_message = "Result deleted successfully!"

    def get\_success\_url(self):

        return reverse("auth:manage\_result")

class VerifyResult(LoginRequiredMixin, View):

    login\_url = 'auth:login'

    def get(self, request):

        return render(request, 'auth/verify\_result.html')

    def post(self, request):

        qs =  request.POST.get('search')

        result = ResultInformation.objects.filter(cert\_no=qs)

        if result:

            return render(request, 'partials/result\_content.html', context={'result':result[0], 'qs':qs})

        else:

            messages.error(request, 'Result not found! try inputting a valid cert\_no')

        return render(request, 'partials/result\_empty.html', {'qs':qs})

class SearchResult(LoginRequiredMixin, View):

    login\_url = 'auth:login'

    def post(self, request):

        qs =  request.POST.get('qs')

        result = ResultInformation.objects.filter(cert\_no=qs)

        if result:

            return render(request, 'auth/verify\_result.html', context={'result':result[0], 'qs':qs})

        else:

            messages.error(request, 'Result not found! try inputting a valid cert\_no')

        return render(request, 'auth/verify\_result.html', {'qs':qs})

**Home Page**

{% extends "base.html" %}

{% load static %}

{% block title %} Home{% endblock %}

{% block head %}

    {% include "partials/head.html" %}

{% endblock %}

{% block body %}

    <div class="container-xxl bg-white p-0">

        <!-- Spinner Start -->

        <div id="spinner" class="show bg-white position-fixed translate-middle w-100 vh-100 top-50 start-50 d-flex align-items-center justify-content-center">

            <div class="spinner-border text-primary" style="width: 3rem; height: 3rem;" role="status">

                <span class="sr-only">Loading...</span>

            </div>

        </div>

        <!-- Spinner End -->

        {% include "partials/nav.html" %}

        <!-- About Start -->

        <div class="container-xxl">

            <div class="container">

                <div class="row g-5 align-items-center">

                    <div class="col-lg-6 wow zoomIn" data-wow-delay="0.1s">

                        <img class="img-fluid" src="{% static 'img/certificate.png' %}">

                    </div>

                    <div class="col-lg-6 wow fadeInUp" data-wow-delay="0.1s">

                        <div class="d-inline-block border rounded-pill text-primary px-4 mb-3">About KASU e-Verify</div>

                        <h2 class="mb-4"> Project</h2>

                        <p class="mb-4">Lorem ipsum dolor sit amet consectetur adipisicing elit. Hic minima optio asperiores ipsam repudiandae .:</p>

                        <div class="row g-3 mb-4">

                            <div class="col-12 d-flex">

                                <div class="flex-shrink-0 btn-lg-square rounded-circle bg-primary">

                                    <i class="fa fa-user-tie text-white"></i>

                                </div>

                                <div class="ms-4">

                                    <h6>vel voluptates</h6>

                                    <!-- <span>CST20ND0558</span> -->

                                </div>

                            </div>

                            <div class="col-12 d-flex">

                                <div class="flex-shrink-0 btn-lg-square rounded-circle bg-primary">

                                    <i class="fa fa-user-tie text-white"></i>

                                </div>

                                <div class="ms-4">

                                    <h6>um aliquid eaque obcaecati</h6>

                                    <!-- <span>CST20HND0558</span> -->

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

        <!-- About End -->

        <!-- Service Start -->

        <div class="container-xxl py-6">

            <div class="container">

                <div class="mx-auto text-center wow fadeInUp" data-wow-delay="0.1s" style="max-width: 600px;">

                    <div class="d-inline-block border rounded-pill text-primary px-4 mb-3">Features of KASU e-Verify</div>

                    <h2 class="mb-5">KASU e-Verify possess some features that makes it flexible</h2>

                </div>

                <div class="row g-4">

                    <div class="col-lg-4 col-md-6 wow fadeInUp" data-wow-delay="0.1s">

                        <div class="service-item rounded h-100">

                            <div class="d-flex justify-content-between">

                                <div class="service-icon">

                                    <i class="fa fa-user-tie fa-2x"></i>

                                </div>

                                <a class="service-btn">

                                    <i class="fa fa-link fa-2x"></i>

                                </a>

                            </div>

                            <div class="p-5">

                                <h5 class="mb-3">Nice User Interface</h5>

                                <span>KASU e-Verify provides a nice user friendly and easy to use user interface.</span>

                            </div>

                        </div>

                    </div>

                    <div class="col-lg-4 col-md-6 wow fadeInUp" data-wow-delay="0.3s">

                        <div class="service-item rounded h-100">

                            <div class="d-flex justify-content-between">

                                <div class="service-icon">

                                    <i class="fa fa-chart-pie fa-2x"></i>

                                </div>

                                <a class="service-btn">

                                    <i class="fa fa-link fa-2x"></i>

                                </a>

                            </div>

                            <div class="p-5">

                                <h5 class="mb-3">Dynamic Search</h5>

                                <span>Fast and no reload on the search page.</span>

                            </div>

                        </div>

                    </div>

                    <div class="col-lg-4 col-md-6 wow fadeInUp" data-wow-delay="0.6s">

                        <div class="service-item rounded h-100">

                            <div class="d-flex justify-content-between">

                                <div class="service-icon">

                                    <i class="fa fa-chart-line fa-2x"></i>

                                </div>

                                <a class="service-btn">

                                    <i class="fa fa-link fa-2x"></i>

                                </a>

                            </div>

                            <div class="p-5">

                                <h5 class="mb-3">Security</h5>

                                <span>KASU e-Verify is a secured application.</span>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

        <!-- Service End -->

       {% include "partials/footer.html" %}

    </div>

    {% include "partials/script.html" %}

{% endblock %}