سند طراحی پروژه

عنوان پروژه: مخزن رزومه (Resume Hub)

توسعه‌دهنده: بهنام شاهمردی (شماره دانشجویی: ۹۳۱۳۰۸۸۰۳۱)

عنوان درس: مهندسی اینترنت

استاد درس: دکتر حبیب‌الله خسروی

تاریخ: ۱۳۹۷/۰۴/۲۵

مشخصات فنی پروژه:

- زبان توسعه: Python 3
- فریم‌ورک توسعه وب: Django 2
- پایگاه‌داده: SQLite 3

آدرس دسترسی آنلاین (از طریق اینترنت) به پروژه:

http://behnamrk.pythonanywhere.com

آدرس صفحه‌ی گیت‌هاب پروژه:

https://github.com/Behnam-RK/Resume-Hub

نحوه‌ی راه‌اندازی و اجرای پروژه:

```
git clone https://github.com/Behnam-RK/Resume-Hub.git

cd Resume-Hub

pip install -r requirements.txt

python manage.py runserver
```

تشریح نحوه‌ی عملکرد پروژه:

این وبسایت برای دسترسی به رزومه‌ی دانشجویان طراحی شده که هر دانشجو می‌تواند رزوره خود را برای دسترسی عموم در سایت قرار بدهد. برای اینکه دانشجو بتواند رزومه‌اش را (همراه با توضیحات (Description)) روی سایت آپلود کند، ابتدا باید در سایت ثبت نام کند. هر دانشجو تنها می‌تواند یک رزومه (و تنها رزومه خودش) را روی سایت قرار دهد. دسترسی به رزومه‌ها برای عموم کاربران آزاد است و نیاز به هیچ مجوز خاص یا ثبت‌نام در سایت ندارد.

همچنین امکان آپلود عکس پروفایل و همچنین تغییر رمز عبور نیز برای کابران پیاده سازی شده است. علاوه‌بر این کاربران می‌توانند برای هر رزومه نظر (Comment) بگذارند، که البته این امکان تنها برای کاربرانی که لاگین کرده‌اند فراهم است.

مستند پروژه (همین فایل) نیز از طریق دکمه‌ی بالا سمت راست تمام صفحات قابل دسترس است اما تنها برای یک کاربر خاص (کاربر از پیش تعریف شده برای مدرس) و برای عموم و حتی کاربران ثبت‌نام کرده نیز قابل دسترسی نیست.

مشخصات کاربران از پیش تعریف شده:

Teacher User:
Username: teacher
Password: changeit

Adminstrator:
Username: admin
Password: changeit

نکته ۱: این کاربران در هنگام اولین اجرای پروژه و ساخت دیتابیس وارد دیتابیس می‌شوند.

نکته ۲: برای دسترسی به قابلیت‌ها و امکانات کاربر Admin به آدرس admin/ مراجعه کنید. (به عنوان مثال Http://localhost:8000/admin)

کد پروژه:

فریم‌ورک Django از معماری MVT بهره می‌برد (Model - View - Template) و برای کار با دیتابیس دارای ORM داخلی می‌باشد و همچنین برای ایجاد Template های داینامیک Template Engine داخلی خود با نام Django Template را در اختیار کاربر قرار می‌دهد.

این پروژه تنها شامل یک Django App به نام hub می‌باشد و تمام عملکردهای وبسایت درون همین App نوشته شده‌اند.

- **فایل models.py**

در این فایل مدل داده‌های برنامه یا همان جدول‌های دیتابیس طراحی‌شـــده‌اند. توجه کنید که خود Django دارای یک مدل User می‌باشـد و ProfileUser در واقع با آن مدل ارتباط یک‌به‌یک دارد. سـایر جزئیات کدها از طریق کامنت‌ها و Docstring ها قابل مشـاهده است.

```
from django.db import models
from django.contrib.auth.models import User


def user_directory_path(instance, filename):
    '''
    for detecting user directory path
    '''
    # file will be uploaded to MEDIA_ROOT/user_<id>/<filename>
    return 'user_{0}/{1}'.format(instance.user.id, filename)


class UserProfile(models.Model):
    '''
    Model for storing resume of user and profile picture and other data...!
    '''
    user = models.OneToOneField(User, on_delete=models.CASCADE)

    # For detect special users (for example teacher user)
    is_special = models.BooleanField(default=False)

    picture = models.ImageField(upload_to=user_directory_path, blank=True)

    resume_file = models.FileField(upload_to=user_directory_path, blank=True)
    description = models.TextField(blank=True)

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)


class Comment(models.Model):
    '''
    Model for storing comments on resumes!
    '''
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    resume = models.ForeignKey(UserProfile, on_delete=models.CASCADE)

    content = models.TextField(blank=True)

    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)


def init_users():
```

```
    '''
    Function for insert specific users in User Model
    '''
    try:
        # Admin User
        admin = User.objects.create_superuser(username='admin', email='admin@resume-hub.com', password='changeit',
first_name='Admin')
        admin_user_profile = UserProfile(user=admin, is_special=True)
        admin_user_profile.save()

        # Teacher User
        teacher_user = User.objects.create_user(username='teacher', email='habiballah_khosravi@yahoo.com',
password='changeit', first_name='Habiballah', last_name='Khosravi')
        teacher_user_profile = UserProfile(user=teacher_user, is_special=True)
        teacher_user_profile.save()

    except:
        pass

init_users()
```

- **فایل forms.py**

این فایل حاوی فرم‌هایی است که قرار است در صفحات نمایش داده شود.

```
from django import forms
from django.contrib.auth.models import User
from hub.models import *


class RegisterForm(forms.ModelForm):
    ''' Register Form '''
    password = forms.CharField(widget=forms.PasswordInput())
    repeat_password = forms.CharField(widget=forms.PasswordInput())
    first_name = forms.CharField(required=True)
    last_name = forms.CharField(required=True)

    class Meta():
        model = User
        fields = ('username', 'password', 'repeat_password', 'first_name', 'last_name', 'email')


class ResumeForm(forms.ModelForm):
    ''' Reseme Form (Dashboard) '''
    class Meta():
        model = UserProfile
        fields = ('resume_file', 'description')


class ChangePasswordForm(forms.Form):
    ''' Change Password Form '''
    current_password = forms.CharField(required=True, widget=forms.PasswordInput())
    new_password = forms.CharField(required=True, widget=forms.PasswordInput())
    repeat_new_password = forms.CharField(required=True, widget=forms.PasswordInput())


class ChangeProfilePicForm(forms.ModelForm):
    ''' Change Profile Picture Form '''
    class Meta():
        model = UserProfile
        fields = ('picture',)


class CommentForm(forms.ModelForm):
    ''' Comment Form '''
    class Meta():
        model = Comment
        fields = ('content',)
        widgets = {
          'content': forms.Textarea(attrs={'rows':2, 'cols':50}),
        }
```

- **فایل views.py**

این فایل حاوی توابعی است که عملکردهای سایت را کنترل می‌کنند. کد بطور کامل کامنت‌گذاری شده و کاملا ماژولار نوشته‌شده‌است و با یک مرور اجمالی قابل فهم است. لازم به ذکر است برای بخش Register و Login از امکانات پیش‌فرض جنگو استفاده‌شده‌است (مانند مدل User و توابع login_user، logout_user، authenticate و ...).

```
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpResponseRedirect
from django.urls import reverse
from django.contrib.auth import authenticate, login as login_user, logout as logout_user
from django.contrib.auth.decorators import login_required
import re
from hub.models import *
from hub.forms import *


# Errors Dictionary!
ERRORS = {
    'invalid_username_or_password': 'Invalid username or password!',
    'account_is_not_activated': 'Your account is not activated!',
    'wrong_current_password': 'Wrong current password!',
    'restricted_section': 'Sorry, You can\'t have access to this section!',
    'not_same_passwords': 'oops! the two passwords you entered are different!'
}

def index(request):
    ''' Index page of website '''
    data = dict()

    # Fetch resumes from database ordered by time (20 most recent resumes first)
    resumes = UserProfile.objects.order_by('updated_at')
    resumes = resumes.reverse()
    if len(resumes) > 20:
        resumes = resumes[:20]

    resumes = [to_dict(resume) for resume in resumes]
    data['resumes'] = resumes

    return render(request, 'hub/index.html', context=data)

def register(request):
    ''' Register! '''
    data = dict()

    registered = False

    if request.user.is_authenticated:
        # check if user is logged-in!
        return redirect(reverse('hub:index'))
```

```
    if request.method == 'POST':
        # create register form object from posted data
        register_form = RegisterForm(data=request.POST)

        if register_form.is_valid():
            password = request.POST['password']
            repeat_password = request.POST['repeat_password']
            if password != repeat_password:
                # password and repeat password are not same error
                return redirect(
                    reverse(
                        'hub:error',
                        kwargs={
                            'error_title': 'not_same_passwords'
                        }
                    )
                )

            # save new user in database
            user = register_form.save()
            user.set_password(user.password)
            user.save()

            registered = True

            # login registered user
            login_user(request, user)

        else:
            # validation error
            error_data = {
                'error_message_alter': register_form.errors.as_ul()
            }
            return render(request, 'hub/error.html', context=error_data)

    else:
        # create empty resume form for GET requests
        register_form = RegisterForm()

    data['register_form'] = register_form
    data['registered'] = registered

    return render(request, 'hub/register.html', context=data)

def login(request):
    ''' Login! '''
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
```

```
        # authenticate user
        user = authenticate(username=username, password=password)

        if user:
            if user.is_active:
                # login user!
                login_user(request, user)

                return redirect(reverse('hub:index'))

            else:
                # not activated account error
                return redirect(
                    reverse(
                        'hub:error',
                        kwargs={
                            'error_title': 'account_is_not_activated'
                        }
                    )
                )

        else:
            # invalid username or password error
            return redirect(
                reverse(
                    'hub:error',
                    kwargs={
                        'error_title': 'invalid_username_or_password'
                    }
                )
            )

    else:
        # GET request!
        return redirect(reverse('hub:index'))

@login_required
def logout(request):
    ''' Logout! '''
    logout_user(request)

    return redirect(reverse('hub:index'))

@login_required
def dashboard(request):
    ''' Dashboard of the logged-in user for uploading resume file '''
    data = dict()

    if request.method == 'POST':
        # create resume form object from posted data
```

```
        resume_form = ResumeForm(data=request.POST)

        if resume_form.is_valid():
            profile = getattr(request.user, 'userprofile', None)
            if profile:
                # user has a UserProfile object
                # save new resume file and description
                profile.description = resume_form.cleaned_data['description']

                if 'resume_file' in request.FILES:
                    profile.resume_file = request.FILES['resume_file']

                profile.save()

                data['current_file'] = profile.resume_file

            else:
                # user does not have a UserProfile object
                #  create a UserProfile object and save resume file and description in it
                profile = resume_form.save(commit=False)
                profile.user = request.user

                if 'resume_file' in request.FILES:
                    profile.resume_file = request.FILES['resume_file']

                profile.save()

                data['current_file'] = profile.resume_file

        else:
            # validation error
            error_data = {
                'error_message_alter': resume_form.errors.as_ul()
            }
            return render(request, 'hub/error.html', context=error_data)

    else:
        # Make empty resume form for GET requests
        profile = getattr(request.user, 'userprofile', None)
        if profile:
            d = {
                'description': profile.description,
                'resume_file': profile.resume_file
            }
            resume_form = ResumeForm(data=d)

            data['current_file'] = profile.resume_file

        else:
            resume_form = ResumeForm()
```

```
    data['resume_form'] = resume_form

    return render(request, 'hub/dashboard.html', context=data)

@login_required
def change_password(request):
    ''' Changes password of the logged-in user '''
    data = dict()

    password_changed = False

    if request.method == 'POST':
        # create profile pic form object from posted data
        change_password_form = ChangePasswordForm(data=request.POST)

        if change_password_form.is_valid():

            change_password_form = change_password_form.cleaned_data

            # get current user
            user = request.user

            if user.check_password(change_password_form['current_password']):
                # current password is correct
                new_password = request.POST['new_password']
                repeat_new_password = request.POST['repeat_new_password']
                if new_password != repeat_new_password:
                    # new password and repeat password are not same
                    return redirect(
                        reverse(
                            'hub:error',
                            kwargs={
                                'error_title': 'not_same_passwords'
                            }
                        )
                    )

                # set new password
                user.set_password(change_password_form['new_password'])
                user.save()

                password_changed = True

            else:
                # current password is incorrect
                return redirect(
                    reverse(
                        'hub:error',
                        kwargs={
```

```
                        'error_title': 'wrong_current_password'
                    }
                )
            )

        else:
            # form data is invalid
            error_data = {
                'error_message_alter': change_password_form.errors.as_ul()
            }
            return render(request, 'hub/error.html', context=error_data)

    else:
        # make and render empty form for GET requests
        change_password_form = ChangePasswordForm()

    data['change_password_form'] = change_password_form
    data['password_changed'] = password_changed

    return render(request, 'hub/change_password.html', context=data)

@login_required
def change_profile_pic(request):
    ''' Changes profile picture of the logged-in user '''
    data = dict()

    # get current user
    user = request.user

    profile_pic_changed = False

    if request.method == 'POST':
        # create profile pic form object from posted data
        change_profile_pic_form = ChangeProfilePicForm(data=request.POST)

        if change_profile_pic_form.is_valid():
            profile = getattr(user, 'userprofile', None)
            if profile:
                # saving profile picture for users that have UserProfile object
                if 'picture' in request.FILES:
                    profile.picture = request.FILES['picture']

                profile.save()

                data['current_picture'] = profile.resume_file

            else:
                # saving profile picture for users that don't have UserProfile object
                profile = change_profile_pic_form.save(commit=False)
                profile.user = user
```

```
                if 'picture' in request.FILES:
                    profile.picture = request.FILES['picture']

                profile.save()

                data['current_picture'] = profile.picture

            profile_pic_changed = True

        else:
            # form is not valid
            error_data = {
                'error_message_alter': change_profile_pic_form.errors.as_ul()
            }
            return render(request, 'hub/error.html', context=error_data)

    else:
        # make and render an empty profile pic form for GET requests
        profile = getattr(user, 'userprofile', None)
        if profile and profile.picture:
            d = {
                'picture': profile.picture
            }
            change_profile_pic_form = ChangeProfilePicForm(data=d)

            data['current_picture'] = profile.picture

        else:
            change_profile_pic_form = ChangeProfilePicForm()

    data['change_profile_pic_form'] = change_profile_pic_form
    data['profile_pic_changed'] = profile_pic_changed

    return render(request, 'hub/change_profile_pic.html', context=data)

def resume(request, user_profile_id):
    ''' Webpage of a specific resume '''
    data = dict()

    # get the resume that is requested
    resume = UserProfile.objects.get(pk=user_profile_id)
    data['resume'] = to_dict(resume)

    # get the comments on that resume
    comments = Comment.objects.filter(resume=resume).order_by('created_at')
    data['comments'] = comments.reverse()

    # create empty comment form
    data['comment_form'] = CommentForm()
```

```
    return render(request, 'hub/resume.html', context=data)

@login_required
def comment(request):
    ''' Commenting on a resume (login required!) '''
    # Get current user
    user = request.user

    if request.method == 'POST':
        # create comment form object from posted data
        comment_form = CommentForm(data=request.POST)

        if comment_form.is_valid():
            # saving the comment
            comment = comment_form.save(commit=False)
            comment.user = user
            resume_id = request.POST['resume_id']
            resume = UserProfile.objects.get(pk=resume_id)
            comment.resume = resume
            comment.save()

            return redirect(
                reverse(
                    'hub:resume',
                    kwargs={
                        'user_profile_id': resume_id
                    }
                )
            )

        else:
            # form is not valid
            error_data = {
                'error_message_alter': comment_form.errors.as_ul()
            }
            return render(request, 'hub/error.html', context=error_data)

    else:
        return redirect(reverse('hub:index'))

def doc(request):
    ''' serve the documentation of the project for special users '''
    data = dict()

    # Getting current user object
    user = request.user

    if not user.is_anonymous:
        # user is logged-in
```

```
        profile = getattr(request.user, 'userprofile', None)
        if profile:
            # user has UserProfile object
            if profile.is_special:
                # user is special!
                return render(request, 'hub/doc.html', context=data)

            else:
                # user is not special!
                return redirect(
                    reverse(
                        'hub:error',
                        kwargs={
                            'error_title': 'restricted_section'
                        }
                    )
                )

        else:
            # user does not have UserProfile object
            return redirect(
                reverse(
                    'hub:error',
                    kwargs={
                        'error_title': 'restricted_section'
                    }
                )
            )

    else:
        # user is anonymous
        return redirect(
            reverse(
                'hub:error',
                kwargs={
                    'error_title': 'restricted_section'
                }
            )
        )

def about(request):
    ''' About! '''
    data = dict()

    return render(request, 'hub/about.html', context=data)

def error(request, error_title):
    ''' webpage for show errors to user '''
    data = dict()
```

```
    # Getting the full message of the error_title from ERRORS dict
    data['error_message'] = ERRORS.get(error_title)

    return render(request, 'hub/error.html', context=data)


def summary(post_body):
    ''' Summerizes a long text with regular expressions '''
    return re.search(r'^.{0,200}[\s]', post_body).group()

def to_dict(resume):
    ''' converts a database object to a dict object '''
    return {
        'id': resume.id,
        'fullname': resume.user.get_full_name(),
        'picture': resume.picture,
        'resume_file': resume.resume_file,
        'summary': summary(resume.description + ' '),
        'description': resume.description,
    }
```

- **فایل urls.py**

این فایل حاوی مسیرهای سایت و توابع معادل آنها در فایل views.py می‌باشد.

```
from django.urls import path
from hub.views import *


app_name = 'hub'

urlpatterns = [
    path('', index, name='index'),
    path('register/', register, name='register'),
    path('login/', login, name='login'),
    path('logout/', logout, name='logout'),
    path('change_password/', change_password, name='change_password'),
    path('dashboard/', dashboard, name='dashboard'),
    path('change_profile_pic/', change_profile_pic, name='change_profile_pic'),
    path('resume/<int:user_profile_id>/', resume, name='resume'),
    path('comment/', comment, name='comment'),
    path('doc/', doc, name='doc'),
    path('about/', about, name='about'),
    path('error/<error_title>', error, name='error')
]
```

- **پوشه‌ی templates**

این پوشه شامل تمام template های پروژه می‌باشد. در بخش Front-End این سایت از فریم‌ورک‌های w3.css و jQuery استفاده شده است. لازم به ذکر است تمام template ها از base.html ارث‌بری می‌کنند.