**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:** ۲۵/۰۴/۱۳۹۷

**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 پیاده‌سازی 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 طراحی­شده­اند. توجه کنید که خود Django دارای یک مدل User می‌باشد و ProfileUser در واقع با آن مدل ارتباط یک‌به‌یک دارد. سایر جزئیات کدها از طریق کامنت­ها و Docstring ها قابل مشاهده 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.