

urls.py

```
from django.contrib import admin
from django.urls import path
from job.views import *
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [path('admin/', admin.site.urls),
path("", index, name="index"),
path('admin_login', admin_login, name="admin_login"),
path('user_login', user_login, name="user_login"),
path('recruiter_login', recruiter_login, name="recruiter_login"),
path('recruiter_home', recruiter_home, name="recruiter_home"),
path('user_signup', user_signup, name="user_signup"),
path('user_home', user_home, name="user_home"),
path('Logout', Logout, name="Logout"),
path('recruiter_signup', recruiter_signup, name="recruiter_signup"),
path('admin_home', admin_home, name="admin_home"),
path('view_users', view_users, name="view_users"),
path('recruiter_pending', recruiter_pending, name="recruiter_pending"),
path('recruiter_accepted', recruiter_accepted, name="recruiter_accepted"),
path('recruiter_rejected', recruiter_rejected, name="recruiter_rejected"),
path('recruiter_all', recruiter_all, name="recruiter_all"),
path('delete_user/<int:pid>', delete_user, name="delete_user"),
path('change_status/<int:pid>', change_status, name="change_status"),
path('delete_recruiter/<int:pid>', delete_recruiter, name="delete_recruiter"),
path('change_passwordadmin', change_passwordadmin, name="change_passwordadmin"),
path('change_passworduser', change_passworduser, name="change_passworduser"),
path('change_passwordrecruiter', change_passwordrecruiter, name="change_passwordrecruiter"),
path('add_job', add_job, name="add_job"),
path('job_list', job_list, name="job_list"),
path('edit_jobdetail/<int:pid>', edit_jobdetail, name="edit_jobdetail"),
path('change_companylogo/<int:pid>', change_companylogo, name="change_companylogo"),
path('latest_jobs', latest_jobs, name="latest_jobs"),
path('user_latestjobs', user_latestjobs, name="user_latestjobs"),
path('job_detail/<int:pid>', job_detail, name="job_detail"),
path('applyforjob/<int:pid>', applyforjob, name="applyforjob"),
path('applied_candidatelist', applied_candidatelist, name="applied_candidatelist"),
path('contact', contact, name="contact"),
]+static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

CODE EXPLANATION:

- ➔ In Django, 'urls.py' is a python module that serves as a central routing configuration file for Web application.
- ➔ It defines the URL patterns and their associated view functions. These URL patterns map to various views or pages this application.
- ➔ Overall, this code defines the structure of web application and how different URLs map to specific views or actions within the application

views.py

```
from django.shortcuts import render, redirect
from .models import *
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from datetime import date

def index(request):
    return render(request, 'index.html')

def admin_login(request):
    error=""
    if request.method=='POST':
        u = request.POST['uname']
        p = request.POST['pwd']
        user = authenticate(username=u,password=p)
        try:
            if user.is_staff:
                login(request,user)
                error="no"
            else:
                error="yes"
        except:
            error="yes"
        d = {'error':error}
        return render(request,'admin_login.html',d)

def user_login(request):
    error=""
    if request.method == "POST":
        u = request.POST['uname'];
        p = request.POST['pwd'];
        user = authenticate(username=u,password=p)
        if user:
            try:
                user1 = StudentUser.objects.get(user=user)
                if user1.type == "student":
                    login(request,user)
                    error="no"
                else:
                    error="yes"
            except:
                error="yes"
            else:
                error="yes"
        d = {'error':error}
        return render(request,'user_login.html',d)

def recruiter_login(request):
    error=""
    if request.method == "POST":
        u = request.POST['uname'];
        p = request.POST['pwd'];
        user = authenticate(username=u,password=p)
        if user:
            try:
                user1 = Recruiter.objects.get(user=user)
                if user1.type == "recruiter" and user1.status!="pending":
                    login(request, user)
                    error="no"
                else:
```

```

error="not"
except:
error="yes"
else:
error="yes"
d = {'error':error}
return render(request,'recruiter_login.html',d)
def recruiter_signup(request):
error = ""
if request.method=='POST':
f = request.POST['fname']
l = request.POST['lname']
i = request.FILES['image']
p = request.POST['pwd']
e = request.POST['email']
con = request.POST['contact']
gen = request.POST['gender']
company = request.POST['company']
try:
user = User.objects.create_user(first_name=f,last_name=l,username=e,password=p)
Recruiter.objects.create(user=user,mobile=con,image=i,gender=gen,company=company,type="recruiter",
status="pending")
error="no"
except:
error="yes"
d = {'error':error}
return render(request,'recruiter_signup.html',d)
def user_home(request):
if not request.user.is_authenticated:
return redirect('user_login')
user = request.user
student = StudentUser.objects.get(user=user)
error=""
if request.method == 'POST':
f = request.POST['fname']
l = request.POST['lname']
con = request.POST['contact']
gen = request.POST['gender']
student.user.first_name = f
student.user.last_name = l
student.mobile = con
student.gender = gen
try:
student.save()
student.user.save()
error = "no"
except:
error = "yes"
try:
i = request.FILES['image']
student.image = i
student.save()
error = "no"
except:
pass
d = {'student':student, 'error':error}
return render(request,'user_home.html',d)
def admin_home(request):
if not request.user.is_authenticated:
return redirect('admin_login')

```

```

rcount = Recruiter.objects.all().count()
scount = StudentUser.objects.all().count()
d = {'rcount':rcount,'scount':scount}
return render(request,'admin_home.html',d)
def recruiter_home(request):
if not request.user.is_authenticated:
return redirect('recruiter_login')
user = request.user
recruiter = Recruiter.objects.get(user=user)
error = ""
if request.method=='POST':
f = request.POST['fname']
l = request.POST['lname']
con = request.POST['contact']
gen = request.POST['gender']
recruiter.user.first_name=f
recruiter.user.last_name=l
recruiter.user.mobile=con
recruiter.user.gender=gen
try:
recruiter.save()
recruiter.user.save()
error="no"
except:
error="yes"
try:
i = request.FILES['image']
recruiter.image = i
recruiter.save()
error="no"
except:
pass
d={'recruiter':recruiter,'error':error}
return render(request,'recruiter_home.html',d)
def Logout(request):
logout(request)
return redirect('index')
def user_signup(request):
error = ""
if request.method=='POST':
f = request.POST['fname']
l = request.POST['lname']
i = request.FILES['image']
p = request.POST['pwd']
e = request.POST['email']
con = request.POST['contact']
gen = request.POST['gender']
try:
user = User.objects.create_user(first_name=f,last_name=l,username=e,password=p)
StudentUser.objects.create(user=user,mobile=con,image=i,gender=gen,type="student")
error="no"
except:
error="yes"
d = {'error':error}
return render(request,'user_signup.html',d)
def view_users(request):
if not request.user.is_authenticated:
return redirect('admin_login')
data = StudentUser.objects.all()
d = {'data':data}

```

```

return render(request,'view_users.html',d)
def latest_jobs(request):
    job = Job.objects.all().order_by('-start_date')
    d = {'job':job}
    return render(request,'latest_jobs.html',d)
def user_latestjobs(request):
    job = Job.objects.all().order_by('-start_date')
    user = request.user
    student = StudentUser.objects.get(user=user)
    data = Apply.objects.filter(student=student)
    li=[]
    for i in data:
        li.append(i.job.id)
    d = {'job':job,'li':li}
    return render(request,'user_latestjobs.html',d)
def delete_user(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    student = User.objects.get(id=pid)
    student.delete()
    return redirect('view_users')
def delete_recruiter(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    recruiter = User.objects.get(id=pid)
    recruiter.delete()
    return redirect('recruiter_all')
def recruiter_pending(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    data = Recruiter.objects.filter(status='pending')
    d = {'data':data}
    return render(request,'recruiter_pending.html',d)
def change_status(request,pid):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    error=""
    recruiter = Recruiter.objects.get(id=pid)
    if request.method=="POST":
        s = request.POST['status']
        recruiter.status=s
    try:
        recruiter.save()
        error="no"
    except:
        error="yes"
    d = {'recruiter':recruiter,'error':error}
    return render(request,'change_status.html',d)
def change_passwordadmin(request):
    if not request.user.is_authenticated:
        return redirect('admin_login')
    error=""
    if request.method=="POST":
        c = request.POST['currentpassword']
        n = request.POST['newpassword']
    try:
        u=User.objects.get(id=request.user.id)
        if u.check_password(c):
            u.set_password(n)
            u.save()

```

```

error="no"
else:
error="not"
except:
error="yes"
d = {'error':error}
return render(request,'change_passwordadmin.html',d)
def change_passworduser(request):
if not request.user.is_authenticated:
return redirect('user_login')
error=""
if request.method=="POST":
c = request.POST['currentpassword']
n = request.POST['newpassword']
try:
u=User.objects.get(id=request.user.id)
if u.check_password(c):
u.set_password(n)
u.save()
error="no"
else:
error="not"
except:
error="yes"
d = {'error':error}
return render(request,'change_passworduser.html',d)
def recruiter_accepted(request):
if not request.user.is_authenticated:
return redirect('admin_login')
data = Recruiter.objects.filter(status='Accept')
d = {'data':data}
return render(request,'recruiter_accepted.html',d)
def recruiter_rejected(request):
if not request.user.is_authenticated:
return redirect('admin_login')
data = Recruiter.objects.filter(status='Reject')
d = {'data':data}
return render(request,'recruiter_rejected.html',d)
def recruiter_all(request):
if not request.user.is_authenticated:
return redirect('admin_login')
data = Recruiter.objects.all()
d = {'data':data}
return render(request,'recruiter_all.html',d)
def change_passwordrecruiter(request):
if not request.user.is_authenticated:
return redirect('recruiter_login')
error=""
if request.method=="POST":
c = request.POST['currentpassword']
n = request.POST['newpassword']
try:
u=User.objects.get(id=request.user.id)
if u.check_password(c):
u.set_password(n)
u.save()
error="no"
else:
error="not"
except:

```

```

error="yes"
d = {'error':error}
return render(request,'change_passwordrecruiter.html',d)
def add_job(request):
if not request.user.is_authenticated:
return redirect('recruiter_login')
error=""
if request.method=='POST':
jt = request.POST['jobtitle']
sd = request.POST['startdate']
ed = request.POST['enddate']
sal = request.POST['salary']
l = request.FILES['logo']
exp = request.POST['experience']
loc = request.POST['location']
skills = request.POST['skills']
des = request.POST['description']
user = request.user
recruiter = Recruiter.objects.get(user=user)
try:
Job.objects.create(recruiter=recruiter,start_date=sd,end_date=ed,title=jt,salary=sal,image=l,description=des,experience=exp,location=loc,skills=skills,creationdate=date.today())
error="no"
except:
error = "yes"
d = {'error':error}
return render(request,'add_job.html',d)
def job_list(request):
if not request.user.is_authenticated:
return redirect('recruiter_login')
user = request.user
recruiter = Recruiter.objects.get(user=user)
job = Job.objects.filter(recruiter=recruiter)
d = {'job':job}
return render(request,'job_list.html',d)
def edit_jobdetail(request,pid):
if not request.user.is_authenticated:
return redirect('recruiter_login')
error=""
job = Job.objects.get(id=pid)
if request.method=='POST':
jt = request.POST['jobtitle']
sd = request.POST['startdate']
ed = request.POST['enddate']
sal = request.POST['salary']
exp = request.POST['experience']
loc = request.POST['location']
skills = request.POST['skills']
des = request.POST['description']
job.title = jt
job.salary = sal
job.experience = exp
job.location = loc
job.skills = skills
job.description = des
try:
job.save()
error="no"
except:
error = "yes"

```

```

if sd:
    try:
        job.start_date = sd
        job.save()
    except:
        pass
    else:
        pass
if ed:
    try:
        job.end_date = ed
        job.save()
    except:
        pass
    else:
        pass
d = {'error':error,'job':job}
return render(request,'edit_jobdetail.html',d)
def change_companylogo(request,pid):
    if not request.user.is_authenticated:
        return redirect('recruiter_login')
    error=""
    job = Job.objects.get(id=pid)
    if request.method=='POST':
        cl = request.FILES['logo']
        job.image = cl
        try:
            job.save()
            error="no"
        except:
            error = "yes"
    d = {'error':error,'job':job}
    return render(request,'change_companylogo.html',d)
def job_detail(request,pid):
    job = Job.objects.get(id=pid)
    d = {'job':job}
    return render(request,'job_detail.html',d)
def applyforjob(request,pid):
    if not request.user.is_authenticated:
        return redirect('user_login')
    error=""
    user = request.user
    student = StudentUser.objects.get(user=user)
    job = Job.objects.get(id=pid)
    date1 = date.today()
    if job.end_date < date1:
        error = "close"
    elif job.start_date > date1:
        error = "notopen"
    else:
        if request.method == 'POST':
            r = request.FILES['resume']
            Apply.objects.create(job=job,student=student,resume=r,applydate=date.today())
            error="done"
            d = {'error':error}
            return render(request,'applyforjob.html',d)
def applied_candidatelist(request):
    if not request.user.is_authenticated:
        return redirect('recruiter_login')
    data = Apply.objects.all()

```



```
d = {'data':data}
return render(request,'applied_candidatelist.html',d)
def contact(request):
return render(request,'contact.html')
```

CODE EXPLANATION

- ➔ views.py is a crucial part of Django applications because it contains the logic that determines how the web application responds to user interactions and requests.
- ➔ Each view function handles a specific aspect of the application, making it modular and organized.
- ➔ It keeps the code maintainable and separates the HTML templates from the application logic.

models.py

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.
class StudentUser(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE)
    mobile = models.CharField(max_length=15,null=True)
    image = models.FileField(null=True)
    gender = models.CharField(max_length=10,null=True)
    type = models.CharField(max_length=15,null=True)
    def __str__(self):
    return self.user.username
class Recruiter(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE)
    mobile = models.CharField(max_length=15,null=True)
    image = models.FileField(null=True)
    gender = models.CharField(max_length=10,null=True)
    company = models.CharField(max_length=100,null=True)
    type = models.CharField(max_length=15,null=True)
    status = models.CharField(max_length=20,null=True)
    def __str__(self):
    return self.user.username
class Job(models.Model):
    recruiter = models.ForeignKey(Recruiter,on_delete=models.CASCADE)
    start_date = models.DateField()end_date = models.DateField()
    title = models.CharField(max_length=100)
    salary = models.FloatField(max_length=20)
    image = models.FileField()
    description = models.CharField(max_length=300)
    experience = models.CharField(max_length=50)
    location = models.CharField(max_length=100)
    skills = models.CharField(max_length=100)
    creationdate = models.DateField()
    def __str__(self):
    return self.title
```

```

class Job(models.Model):
    recruiter = models.ForeignKey(Recruiter,on_delete=models.CASCADE)
    start_date = models.DateField()
    end_date = models.DateField()
    title = models.CharField(max_length=100)
    salary = models.FloatField(max_length=20)
    image = models.FileField()
    description = models.CharField(max_length=300)
    experience = models.CharField(max_length=50)
    location = models.CharField(max_length=100)
    skills = models.CharField(max_length=100)
    creationdate = models.DateField()
    def __str__(self):
    return self.title
class Apply(models.Model):
    job = models.ForeignKey(Job,on_delete=models.CASCADE)
    student = models.ForeignKey(StudentUser,on_delete=models.CASCADE)
    resume = models.FileField(null=True)
    applydate = models.DateField()
    def __str__(self):
    return self.id

```

CODE EXPLANATION:

- ➔ ‘models.py’ is crucial for defining the data structure of Django application.
- ➔ It allows to create, retrieve, update and delete data in the database, and it forms the foundation for data-driven web applications.
- ➔ These models define the database tables and relationships for storing user information, job postings, and job applications.

navigation.html

```

<html>
<head>
{% load static %}
<title>Job Portal</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css">
<script src="https://cdn.jsdelivr.net/npm/jquery@3.6.4/dist/jquery.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"></script>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@4.7.0/css/font-awesome.min.css">
<style>
.nav-link {
color : white !important;
margin-right: 4px;}
.nav-link:hover {
color : yellow !important;
font-weight: bold;

```

```

background-color: red;
border-radius: 10px;}
.navbar-brand:hover {
color: yellow !important;
}
</style></head><body>
<div class="container">
<div class="row">
<div class="col-sm-3">
</div>
<div class="col-sm-6 mt-5 text-center">
<span style="font-weight: bold; font-size: 30px">
<span style="color : red">Online</span>
<span style="color : blue">Job</span>
<span style="color : maroon">Portal</span>
<span style="color : green">System</span></span></div>
<div class="col-sm-3 mt-5" style="padding-left: 10%;">
<span style="font-weight: bold; font-size: 30px">
<span><i class="fa fa-twitter pl-1"></i></span>
<span><i class="fa fa-instagram pl-1"></i></span>
<span><i class="fa fa-linkedin pl-1"></i></span>
<span><i class="fa fa-telegram pl-1"></i></span></span></div></div></div>
<nav class="navbar navbar-expand-sm" style="background-color: maroon">
<a class="navbar-brand text-white pl-5 font-weight-bold" style="font-family: 'Monotype Corsiva';font-size:30px">Job Portal</a>
<ul class="navbar-nav">
<li class="nav-item">
<a class="nav-link" href="{% url 'index' %}" style="margin-left : 280px"><i class="fa fa-home mr-1"></i>Home</a></li>
<li class="nav-item">
<a class="nav-link" href="{% url 'latest_jobs' %}"><i class="fa fa-users mr-1"></i>Latest Jobs</a></li>
<li class="nav-item">
<a class="nav-link" href="{% url 'user_login' %}"><i class="fa fa-user-plus mr-1"></i>User Login</a></li>
<li class="nav-item">
<a class="nav-link" href="{% url 'recruiter_login' %}"><i class="fa fa-sign-language mr-1"></i>Recruiter Login</a></li>
<li class="nav-item">
<a class="nav-link" href="{% url 'admin_login' %}"><i class="fa fa-sign-in mr-1"></i>AdminLogin</a></li>
<li class="nav-item">
<a class="nav-link" href="{% url 'contact' %}"><i class="fa fa-sign-in mr-1"></i>Contact</a>
</li></ul></nav></body></html>

```

CODE EXPLANATION:

- ➔ It is an HTML template for a web page in Django application, which appears to be included in other pages to provide a consistent navigation bar.
- ➔ It helps ensure a uniform and user-friendly user interface by reusing this navigation bar on various pages, reducing the code, and making it easier to update the navigation across the site.
- ➔ ‘{% load static %}’ which means it loads the static files such as CSS and image files, allowing you to use them in the template.

admin_home.html

```
{% load static %}
{% block body %}
{% include 'admin_navigation.html' %}
<style>
.mydiv:hover {
box-shadow: 0 0 20px 0 rgba(0,0,0,0.3);
transform: translateY(-20px); }</style>
<marquee style="font-family: 'Monotype Corsiva'; font-size: 25px; color: darkred;
background-color: yellow; font-weight: bold; margin-top: 1%;"> Contact
Us</marquee><hr>
<div class="container mt-5">
<br><br>
<div class="row mt-5">
<div class="col-sm-6 mydiv">
<h4 class="text-center"><i class="fa fa-user mr-2"></i>Total Recruiters</h4>
<p class="text-center font-weight-bold">{{ rcount }}</p></div>
<div class="col-sm-6 mydiv">
<h4 class="text-center"><i class="fa fa-user mr-2"></i>Total Users</h4>
<p class="text-center font-weight-bold">{{ scount }}</p></div></div></div>
{% endblock %}
```

CODE EXPLANATION:

- ➔ It is a template which is designed for displaying contact and summary information.
- ➔ It utilizes dynamic content to show the total count of recruiters and users, making it a useful tool for administrators or users to quickly access relevant information.
- ➔ The custom CSS styles enhance the visual presentation of the page, creating an animation effect when hovering over elements with the class 'mydiv'.

admin_login

```
{% load static %}
{% block body %}
{% include 'navigation.html' %}
<marquee style="font-family: 'Monotype Corsiva'; font-size: 25px; color: white; background-
color: blue; font-weight: bold; margin-top: 1%;">Admin Login</marquee>
<hr>
<form class="container mt-5" method="POST">
{% csrf_token %}
<label><b>UserName</b></label>
<input type="text" class="form-control" name="uname" placeholder="Enter UserName">
<label><b>Password</b></label>
<input type="password" class="form-control" name="pwd" placeholder="Enter Password">
```

```

<input type="submit" class="btn-primary" style="margin-top: 2%; color: white; background-
color: blue; height: 40px; font-size: 20px"></form>
{% ifequal error "no" %}
<script>
alert("Login successful");
window.location=('{% url 'admin_home' %}')
</script>
{% endifequal %}
{% ifequal error "yes" %}
<script>
alert("Invalid Login credentials ");
</script>
{% endifequal %}
{% endblock %}

```

CODE EXPLANATION:

- ➔ This template is designed for the admin login page in a Django application, where administrators can enter their credentials to access the admin panel.
- ➔ It includes the navigation menu for site-wide consistency.
- ➔ The use of conditional statements in this code allows for feedback to be displayed based on the success or failure of the login attempt.
- ➔ The CSRF token ensures the security of the form submission.

job_details.html

```

{% load static %}
{% block body %}
{% include 'user_navigation.html' %}
<style>
tr{text-align: center}</style>
<marquee style="font-family: 'Monotype Corsiva'; font-size: 25px; color: blue; background-color:
lightpink; font-weight: bold; margin-top: 1%">Job Details</marquee><hr>
<div class="container shadow-lg py-2 mb-2">
<div class="row">
<div class="col-md-4">
</div>
<div class="col-md-6">
<h2>{% job.title %}</h2>
<p>{% job.recruiter.company %} <em><a href="{% url 'user_latestjobs' %}">(View All
Jobs)</a></em></p>
<a href="#" class="mr-2"><i class="fa fa-map-marker mr-2"></i>{% job.location %}</a>
<a href="#"><i class="fa fa-calendar mr-2"></i>{% job.creationdate %}</a><br>
<strong><i class="fa fa-money mr-2 mt-2"></i><i class="fa fa-inr mr-2 mt-
2"></i>{% job.salary %}</strong>
<div class="mt-4">
<a href="{% url 'applyforjob' job.id %}" class="btn btn-danger"
style="width:200px;height:40p">Apply for this Job</a></div></div></div>
<div class="mt-3">
<h4>Overview</h4>
<p>{% job.description %}</p>

```

```

<h4>Required Experience</h4>
<p>{{job.experience}} Yrs</p>
<h4>Skills Required</h4>
<p>{{job.skills}}</p>
<h4>Start Date for Applying</h4>
<p>{{job.start_date}}</p>
<h4>Last Date for Applying</h4>
<p>{{job.end_date}}</p></div></div>
{% endblock %}

```

CODE EXPLANATION:

- ➔ This template is designed to provide users with detailed information about a specific job, including its title, recruiter, location, salary and other job-related details.
- ➔ It provides all the essential information that job seekers need to decide whether to apply for a particular job.
- ➔ It enables job applications by linking to the application page.

latest_jobs.html

```

{% load static %}
{% block body %}
{% include 'navigation.html' %}
<style>
tr{text-align: center} </style>
<marquee style="font-family: 'Monotype Corsiva'; font-size: 25px; color: darkred; background-color: bisque; font-weight: bold; margin-top: 1%"> Latest Jobs List</marquee><hr>
<div class="container">
<table class="table table-hover " id="example">
<thead>
<tr style="background-color: #000080; color: white;">
<th>S.NO.</th>
<th>Company Name</th>
<th>Job Title</th>
<th>Location</th>
<th>Start Date</th>
<th>End Date</th>
<th>Action</th></tr></thead><tbody>
{% for i in job %}
<tr style="background-color: #DEB887">
<td>{{forloop.counter}}</td>
<td>{{i.recruiter.company}}</td>
<td>{{i.title}}</td>
<td>{{i.location}}</td>
<td>{{i.start_date}}</td>
<td>{{i.end_date}}</td>
<td><a href="{% url 'user_login' %}" class="btn btn-success btn-sm">Login to Apply</a></td></tr>
{% endfor %}
</tbody></table></div>
{% endblock %}

```

CODE EXPLANATION:

- ➔ This template is to provide a user-friendly list of the latest job postings.
- ➔ Job seekers can quickly access information about job titles, companies and application deadlines.
- ➔ The template encourages users to log in (or sign up) to apply for jobs by providing a “Login to Apply” button.
- ➔ It enhances the user experience by presenting the latest job listings in a clear and organized manner.

recruiter signup

```
{% load static %}
{% block body %}
{% include 'navigation.html' %}
<script>
function checkpass() {
if(document.signup.pwd.value!=document.signup.cpwd.value) {
alert('Password and Confirm Password are not same');
document.signup.cpwd.focus();
return false; }
return true; }
</script>
<marquee style="font-family: 'Monotype Corsiva'; font-size: 25px; color: forestgreen; background-color: deepskyblue; font-weight: bold; margin-top: 1%;">Recruiter SignUp </marquee><hr>
<form class="container mt-5" onsubmit="return checkpass();" name="signup" method="POST"
enctype="multipart/form-data">
{% csrf_token %}
<div class="form-row">
<div class="form-group col-md-6">
<label>First Name</label>
<input type="text" class="form-control" name="fname" placeholder="Enter First Name" required>
</div>
<div class="form-group col-md-6">
<label>Last Name</label>
<input type="text" class="form-control" name="lname" placeholder="Enter Last Name"></div></div>
<div class="form-row">
<div class="form-group col-md-12">
<label>Contact</label>
<input type="text" class="form-control" name="contact" placeholder="Enter Contact Number"
required>
</div>
<div class="form-group col-md-6">
<label>Company</label>
<input type="text" class="form-control" name="company" placeholder="Enter Your Company Name"
required>
</div>
</div>
<div class="form-row">
<div class="form-group col-md-12">
<label>Email ID</label>
<input type="text" class="form-control" name="email" placeholder="Enter Email Address" required>
```

```

</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<label>Password</label>
<input type="password" class="form-control" name="pwd" placeholder="Enter Password" required>
</div>
<div class="form-group col-md-6">
<label>Confirm Password</label>
<input type="password" class="form-control" name="cpwd" placeholder="Enter Confirm Password"
required>
</div>
</div>
<div class="form-row">
<div class="form-group col-md-6">
<label>Gender</label>
<div style="border: 1px solid lightgrey; padding: 6px; border-radius: 6px;">
<div class="custom-control custom-radio custom-control-inline">
<input type="radio" id="customRadioInline1" name="gender" class="custom-control-input"
value="Male">
<label class="custom-control-label" for="customRadioInline1">Male</label></div>
<div class="custom-control custom-radio custom-control-inline">
<input type="radio" id="customRadioInline2" name="gender" class="custom-control-input"
value="Female">
<label class="custom-control-label" for="customRadioInline2">Female</label></div></div></div>
<div class="form-group col-md-6">
<label>Image</label>
<input type="file" class="form-control" name="image" required></div></div>
<input type="submit" value="Submit" class="btn btn-primary my-3" style="width: 200px">
</form>
{% include 'footer.html' %}
{% ifequal error "no" %}
<script>
alert("Signup successful");
window.location=('{% url 'recruiter_login' %}')
</script>
{% endifequal %}
{% ifequal error "yes" %}
<script>
alert("Something went wrong try again");
</script>
{% endifequal %}
{% endblock %}

```

CODE EXPLANATION:

- ➔ This template is designed for recruiter sign-up in a web application.
- ➔ It collects necessary information, including personal details, contact information, and authentication credentials.
- ➔ The form includes client-side validation to ensure the password and confirm password fields match, enhancing user experience.
- ➔ The error handling ensures that users receive feedback about the sign-up process, whether it's successful or if something went wrong.

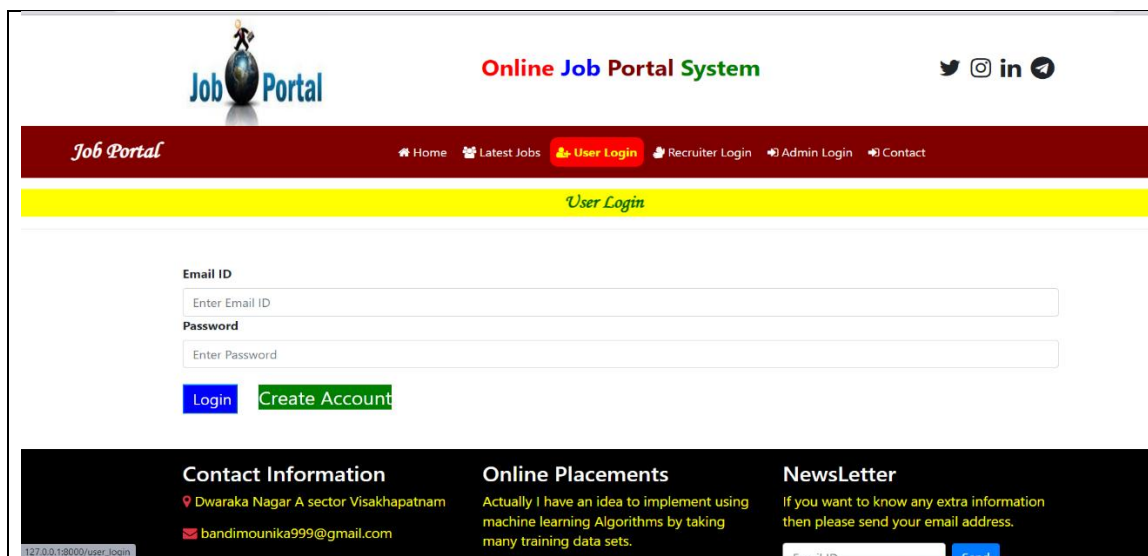
view_users.html


```
{% load static %}
{% block body %}
{% include 'admin_navigation.html' %}
<marquee style="font-family: 'Monotype Corsiva'; font-size: 25px; color: darkgreen; background-color:
yellow; font-weight: bold; margin-top: 1%;">View All Student Users
</marquee>
<hr>
<div class="container">
<table class="table table-hover " id="example">
<thead>
<tr style="background-color: #000080; color: white;">
<th>S.NO.</th>
<th>Full Name</th>
<th>Email ID</th>
<th>Contact</th>
<th>Gender</th>
<th>Image</th>
<th>Action</th>
</tr>
</thead>
<tbody>
{% for i in data %}
<tr style="background-color: #DEB887">
<td>{{ forloop.counter }}</td>
<td>{{ i.user.first_name }} {{ i.user.last_name }}</td>
<td>{{ i.user.username }}</td>
<td>{{ i.mobile }}</td>
<td>{{ i.gender }}</td>
<td></td>
<td><a href="{% url 'delete_user' i.user.id %}" class="btn btn-danger btn-sm" onclick="return
confirm('Are you sure want to delete?')>Delete</a></td>
</tr>
{% endfor %}
</tbody>
</table>
</div>
{% endblock %}
```

CODE EXPLANATION:





- ➔ This template is to provide administrators with a user-friendly interface to view and manage student user data.
- ➔ It allows administrators to quickly student users, view their details, and delete them if necessary.
- ➔ By displaying user information in a table format, it organizes the data and makes it easily accessible for administrators.

PHOTOS





Online Job Portal System



Job Portal

[Home](#) [Latest Jobs](#) [User Login](#) [Recruiter Login](#) [Admin Login](#) [Contact](#)

Recruiter Login

UserName

Enter UserName


Password

Enter Password





Submit

Create Account

127.0.0.1:8000/recruiter_login



Online Job Portal System



Job Portal

[Home](#) [Latest Jobs](#) [User Login](#) [Recruiter Login](#) [Admin Login](#) [Contact](#)

Admin Login

UserName


Enter UserName

Password





Enter Password

Submit

127.0.0.1:8000/admin_login




Online Job Portal System




Job Portal

[Home](#) [Latest Jobs](#) [User Login](#) [Recruiter Login](#) [Admin Login](#) [Contact](#)


Contact Us

 Contact


+91 9087675445

 Email ID

bandimounika999@gmail.com


 Address

Vizag





 Office Timings

10:00 A.M. to 7:00 P.M.

127.0.0.1:8000/contact



Online Job Portal System

Job Portal

[Home](#)
[Recruiters](#)
[View Users](#)
[Change Password](#)
[Logout](#)

[Copy](#)
[Excel](#)
[CSV](#)
[PDF](#)

Pending




Accepted

Rejected


All Recruiter

View Recruiters - All





Search:

S.NO.	Full Name	Email ID	Contact	Gender	Company	Image	Status	Action
1	BETHAPUDI PRAKASH	prakash@gmail.com	8688477420	Male	TCS		Accept	Change Status Delete
2	BANDI SUNITHA	sunitha@gmail.com	9398880287	Female	AMAZON		Accept	Change Status Delete
3	ABDUL KALAM	abdul@gmail.com	9876012345	Male	INFOSYS		Accept	Change Status Delete

127.0.0.1:8000/recruiter_all#



Online Job Portal System









Job Portal


[Home](#)
[Recruiters](#)
[View Users](#)
[Change Password](#)
[Logout](#)

View All Student Users





Search:

S.NO.	Full Name	Email ID	Contact	Gender	Image	Action
1	BANDI SAI	sai@gmail.com	7890654321	Male		Delete
2	PALAVALASA HARI	hari@gmail.com	7896543210	Male		Delete
3	SAMBANA HARSHITHA	harshi@gmail.com	8688477420	Female		Delete

127.0.0.1:8000/view_users



Online Job Portal System

Job Portal

[Home](#)
[Recruiters](#)
[View Users](#)
[Change Password](#)
[Logout](#)

Change Password

Current Password

.....


New Password

....





Confirm New Password

....

Submit




Online Job Portal System



Job Portal



[Home](#) [Job List](#) [Change Password](#) [Logout](#) PALAVALASA HARI


Job Details



BACKEND DEVELOPER

INFOSYS [\(View All Jobs\)](#)

 Chennai  Oct. 14, 2023

 ₹ 400000.0

[Apply for this Job](#)

Overview

24 vacancies

Required Experience

1 Yrs





Skills Required

Java, Spring, Mysql

127.0.0.1:8000/user_latestjobs



Online Job Portal System



Job Portal

[Home](#) [Add Job](#) [Job List](#) [Candidate Applied](#) [Change Password](#) [Logout](#) ABDUL KALAM

Add Job Details

Job Title

Java Developer

Start Date

12-10-2023

End Date

19-12-2023

Salary(Per Month)

60000.0

Company Logo

[Choose file](#) c5.jpeg

Experience(in years)


2

Company Location





Skills

Java, Spring, Mysql

127.0.0.1:8000/add_job



Online Job Portal System



Job Portal

[Home](#) [Add Job](#) [Job List](#) [Candidate Applied](#) [Change Password](#) [Logout](#) ABDUL KALAM

Applied Candidate List

S.NO.	FullName	EmailID	Contact No.	Job Title	Company	AppliedDate	Resume
1	BANDI SAI	sai@gmail.com	7890654321	TESTER	INFOSYS	Oct. 14, 2023	Download Resume
2	B MOUNIKA	mounika@gmail.com	9012345678	TESTER	INFOSYS	Oct. 14, 2023	Download Resume
3	PALAVALASA HARI	hari@gmail.com	7896543210	TESTER	INFOSYS	Oct. 14, 2023	Download Resume

127.0.0.1:8000/applied_candidateList