**Core functionalities**

**Creating administrative functions**

from django.contrib import admin

from .models import \*

# Register your models here.

admin.site.register(Admin\_Helath\_CSV)

admin.site.register(Doctor)

admin.site.register(Patient)

admin.site.register(Feedback)

admin.site.register(Search\_Data)

**Explanation**

This code imports the Django admin tools along with all the defined models in your project. It then registers several models—Admin\_Helath\_CSV, Doctor, Patient, Feedback, and Search\_Data—with the Django admin site. By doing so, these models become accessible through Django’s built-in admin interface, which is a user-friendly control panel. This setup allows administrators to easily add, modify, or delete data related to health records, doctors, patients, and feedback. In simple terms, the code creates an easy-to-use dashboard for managing your application's data without needing to write complex code.

The base code for managing the entire system

from django.contrib import messages

from django.contrib.auth.decorators import login\_required

from django.shortcuts import render, redirect

import datetime

from sklearn.ensemble import GradientBoostingClassifier

from .forms import DoctorForm

from .models import \*

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

sns.set\_style('darkgrid')

from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from django.http import HttpResponse

# Create your views here.

def Home(request):

return render(request,'carousel.html')

def Admin\_Home(request):

dis = Search\_Data.objects.all()

pat = Patient.objects.all()

doc = Doctor.objects.all()

feed = Feedback.objects.all()

d = {'dis':dis.count(),'pat':pat.count(),'doc':doc.count(),'feed':feed.count()}

return render(request,'admin\_home.html',d)

@login\_required(login\_url="login")

def assign\_status(request,pid):

doctor = Doctor.objects.get(id=pid)

if doctor.status == 1:

doctor.status = 2

messages.success(request, 'Selected doctor are successfully withdraw his approval.')

else:

doctor.status = 1

messages.success(request, 'Selected doctor are successfully approved.')

doctor.save()

return redirect('view\_doctor')

@login\_required(login\_url="login")

def User\_Home(request):

return render(request,'patient\_home.html')

@login\_required(login\_url="login")

def Doctor\_Home(request):

return render(request,'doctor\_home.html')

def About(request):

return render(request,'about.html')

def Contact(request):

return render(request,'contact.html')

def Gallery(request):

return render(request,'gallery.html')

def Login\_User(request):

error = ""

if request.method == "POST":

u = request.POST['uname']

p = request.POST['pwd']

user = authenticate(username=u, password=p)

sign = ""

if user:

try:

sign = Patient.objects.get(user=user)

except:

pass

if sign:

login(request, user)

error = "pat1"

else:

pure=False

try:

pure = Doctor.objects.get(status=1,user=user)

except:

pass

if pure:

login(request, user)

error = "pat2"

else:

login(request, user)

error="notmember"

else:

error="not"

d = {'error': error}

return render(request, 'login.html', d)

def Login\_admin(request):

error = ""

if request.method == "POST":

u = request.POST['uname']

p = request.POST['pwd']

user = authenticate(username=u, password=p)

if user.is\_staff:

login(request, user)

error="pat"

else:

error="not"

d = {'error': error}

return render(request, 'admin\_login.html', d)

def Signup\_User(request):

error = ""

if request.method == 'POST':

f = request.POST['fname']

l = request.POST['lname']

u = request.POST['uname']

e = request.POST['email']

p = request.POST['pwd']

d = request.POST['dob']

con = request.POST['contact']

add = request.POST['add']

type = request.POST['type']

im = request.FILES['image']

dat = datetime.date.today()

user = User.objects.create\_user(email=e, username=u, password=p, first\_name=f,last\_name=l)

if type == "Patient":

Patient.objects.create(user=user,contact=con,address=add,image=im,dob=d)

else:

Doctor.objects.create(dob=d,image=im,user=user,contact=con,address=add,status=2)

error = "create"

d = {'error':error}

return render(request,'register.html',d)

def Logout(request):

logout(request)

return redirect('home')

@login\_required(login\_url="login")

def Change\_Password(request):

sign = 0

user = User.objects.get(username=request.user.username)

error = ""

if not request.user.is\_staff:

try:

sign = Patient.objects.get(user=user)

if sign:

error = "pat"

except:

sign = Doctor.objects.get(user=user)

terror = ""

if request.method=="POST":

n = request.POST['pwd1']

c = request.POST['pwd2']

o = request.POST['pwd3']

if c == n:

u = User.objects.get(username\_\_exact=request.user.username)

u.set\_password(n)

u.save()

terror = "yes"

else:

terror = "not"

d = {'error':error,'terror':terror,'data':sign}

return render(request,'change\_password.html',d)

def preprocess\_inputs(df, scaler):

df = df.copy()

# Split df into X and y

y = df['target'].copy()

X = df.drop('target', axis=1).copy()

X = pd.DataFrame(scaler.fit\_transform(X), columns=X.columns)

return X, y

def prdict\_heart\_disease(list\_data):

csv\_file = Admin\_Helath\_CSV.objects.get(id=1)

df = pd.read\_csv(csv\_file.csv\_file)

X = df[['age','sex','cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']]

y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.8, random\_state=0)

nn\_model = GradientBoostingClassifier(n\_estimators=100,learning\_rate=1.0,max\_depth=1, random\_state=0)

nn\_model.fit(X\_train, y\_train)

pred = nn\_model.predict([list\_data])

print("Neural Network Accuracy: {:.2f}%".format(nn\_model.score(X\_test, y\_test) \* 100))

print("Prdicted Value is : ", format(pred))

dataframe = str(df.head())

return (nn\_model.score(X\_test, y\_test) \* 100),(pred)

@login\_required(login\_url="login")

def add\_doctor(request,pid=None):

doctor = None

if pid:

doctor = Doctor.objects.get(id=pid)

if request.method == "POST":

form = DoctorForm(request.POST, request.FILES, instance = doctor)

if form.is\_valid():

new\_doc = form.save()

new\_doc.status = 1

if not pid:

user = User.objects.create\_user(password=request.POST['password'], username=request.POST['username'], first\_name=request.POST['first\_name'], last\_name=request.POST['last\_name'])

new\_doc.user = user

new\_doc.save()

return redirect('view\_doctor')

d = {"doctor": doctor}

return render(request, 'add\_doctor.html', d)

@login\_required(login\_url="login")

def add\_heartdetail(request):

if request.method == "POST":

# list\_data = [57, 0, 1, 130, 236, 0, 0, 174, 0, 0.0, 1, 1, 2]

list\_data = []

value\_dict = eval(str(request.POST)[12:-1])

count = 0

for key,value in value\_dict.items():

if count == 0:

count =1

continue

if key == "sex" and value[0] == "Male" or value[0] == 'male' or value[0]=='m' or value[0] == 'M':

list\_data.append(0)

continue

elif key == "sex":

list\_data.append(1)

continue

list\_data.append(value[0])

# list\_data = [57, 0, 1, 130, 236, 0, 0, 174, 0, 0.0, 1, 1, 2]

accuracy,pred = prdict\_heart\_disease(list\_data)

patient = Patient.objects.get(user=request.user)

Search\_Data.objects.create(patient=patient, prediction\_accuracy=accuracy, result=pred[0], values\_list=list\_data)

rem = int(pred[0])

print("Result = ",rem)

if pred[0] == 0:

pred = "<span style='color:green'>You are healthy</span>"

else:

pred = "<span style='color:red'>You are Unhealthy, Need to Checkup.</span>"

return redirect('predict\_desease', str(rem), str(accuracy))

return render(request, 'add\_heartdetail.html')

@login\_required(login\_url="login")

def predict\_desease(request, pred, accuracy):

doctor = Doctor.objects.filter(address\_\_icontains=Patient.objects.get(user=request.user).address)

d = {'pred': pred, 'accuracy':accuracy, 'doctor':doctor}

return render(request, 'predict\_disease.html', d)

@login\_required(login\_url="login")

def view\_search\_pat(request):

doc = None

try:

doc = Doctor.objects.get(user=request.user)

data = Search\_Data.objects.filter(patient\_\_address\_\_icontains=doc.address).order\_by('-id')

except:

try:

doc = Patient.objects.get(user=request.user)

data = Search\_Data.objects.filter(patient=doc).order\_by('-id')

except:

data = Search\_Data.objects.all().order\_by('-id')

return render(request,'view\_search\_pat.html',{'data':data})

@login\_required(login\_url="login")

def delete\_doctor(request,pid):

doc = Doctor.objects.get(id=pid)

doc.delete()

return redirect('view\_doctor')

@login\_required(login\_url="login")

def delete\_feedback(request,pid):

doc = Feedback.objects.get(id=pid)

doc.delete()

return redirect('view\_feedback')

@login\_required(login\_url="login")

def delete\_patient(request,pid):

doc = Patient.objects.get(id=pid)

doc.delete()

return redirect('view\_patient')

@login\_required(login\_url="login")

def delete\_searched(request,pid):

doc = Search\_Data.objects.get(id=pid)

doc.delete()

return redirect('view\_search\_pat')

@login\_required(login\_url="login")

def View\_Doctor(request):

doc = Doctor.objects.all()

d = {'doc':doc}

return render(request,'view\_doctor.html',d)

@login\_required(login\_url="login")

def View\_Patient(request):

patient = Patient.objects.all()

d = {'patient':patient}

return render(request,'view\_patient.html',d)

@login\_required(login\_url="login")

def View\_Feedback(request):

dis = Feedback.objects.all()

d = {'dis':dis}

return render(request,'view\_feedback.html',d)

@login\_required(login\_url="login")

def View\_My\_Detail(request):

terror = ""

user = User.objects.get(id=request.user.id)

error = ""

try:

sign = Patient.objects.get(user=user)

error = "pat"

except:

sign = Doctor.objects.get(user=user)

d = {'error': error,'pro':sign}

return render(request,'profile\_doctor.html',d)

@login\_required(login\_url="login")

def Edit\_Doctor(request,pid):

doc = Doctor.objects.get(id=pid)

error = ""

# type = Type.objects.all()

if request.method == 'POST':

f = request.POST['fname']

l = request.POST['lname']

e = request.POST['email']

con = request.POST['contact']

add = request.POST['add']

cat = request.POST['type']

try:

im = request.FILES['image']

doc.image=im

doc.save()

except:

pass

dat = datetime.date.today()

doc.user.first\_name = f

doc.user.last\_name = l

doc.user.email = e

doc.contact = con

doc.category = cat

doc.address = add

doc.user.save()

doc.save()

error = "create"

d = {'error':error,'doc':doc,'type':type}

return render(request,'edit\_doctor.html',d)

@login\_required(login\_url="login")

def Edit\_My\_deatail(request):

terror = ""

print("Hii welvome")

user = User.objects.get(id=request.user.id)

error = ""

# type = Type.objects.all()

try:

sign = Patient.objects.get(user=user)

error = "pat"

except:

sign = Doctor.objects.get(user=user)

if request.method == 'POST':

f = request.POST['fname']

l = request.POST['lname']

e = request.POST['email']

con = request.POST['contact']

add = request.POST['add']

try:

im = request.FILES['image']

sign.image = im

sign.save()

except:

pass

to1 = datetime.date.today()

sign.user.first\_name = f

sign.user.last\_name = l

sign.user.email = e

sign.contact = con

if error != "pat":

cat = request.POST['type']

sign.category = cat

sign.save()

sign.address = add

sign.user.save()

sign.save()

terror = "create"

d = {'error':error,'terror':terror,'doc':sign}

return render(request,'edit\_profile.html',d)

@login\_required(login\_url='login')

def sent\_feedback(request):

terror = None

if request.method == "POST":

username = request.POST['uname']

message = request.POST['msg']

username = User.objects.get(username=username)

Feedback.objects.create(user=username, messages=message)

terror = "create"

return render(request, 'sent\_feedback.html',{'terror':terror})

**Explanation**

This code is a collection of functions for a Django web application designed to manage a healthcare system. It defines views that render various pages like home, about, contact, and gallery to make the site user-friendly. The admin dashboard aggregates key data such as the counts of searches, patients, doctors, and feedback for easy monitoring. User authentication is handled by functions that allow patients, doctors, and admins to log in and sign up securely. Once logged in, users are directed to personalized home pages based on their role, ensuring a tailored experience. There is a feature that lets administrators toggle a doctor’s approval status, thereby controlling access to the system. Users can update their profiles and change their passwords using dedicated functions, which helps maintain current information. The code also integrates a machine learning model that predicts heart disease by analyzing user-submitted health data. This model is trained with historical data, and its prediction accuracy is calculated and displayed. Finally, the system records each search and prediction, and it even recommends local doctors based on the user's address.

Creating database models and ensuring a successful

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

from .choices import DOCTOR\_STATUS

class Patient(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE, null=True)

contact = models.CharField(max\_length=100, null=True)

address = models.CharField(max\_length=100, null=True)

dob = models.DateField(null=True)

image = models.FileField(null=True)

def \_\_str\_\_(self):

return self.user.username

class Doctor(models.Model):

status = models.IntegerField(DOCTOR\_STATUS, null=True)

user = models.ForeignKey(User, on\_delete=models.CASCADE, null=True)

contact = models.CharField(max\_length=100, null=True)

address = models.CharField(max\_length=100, null=True)

category = models.CharField(max\_length=100, null=True)

doj = models.DateField(null=True)

dob = models.DateField(null=True)

image = models.FileField(null=True)

def \_\_str\_\_(self):

return self.user.username

class Admin\_Helath\_CSV(models.Model):

name = models.CharField(max\_length=100, null=True)

csv\_file = models.FileField(null=True, blank=True)

def \_\_str\_\_(self):

return self.name

class Search\_Data(models.Model):

patient = models.ForeignKey(Patient, on\_delete=models.CASCADE, null=True)

prediction\_accuracy = models.CharField(max\_length=100,null=True,blank=True)

result = models.CharField(max\_length=100,null=True,blank=True)

values\_list = models.CharField(max\_length=100,null=True,blank=True)

created = models.DateTimeField(auto\_now=True,null=True)

def \_\_str\_\_(self):

return self.patient.user.username

class Feedback(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE, null=True)

messages = models.TextField(null=True)

date = models.DateField(auto\_now=True)

def \_\_str\_\_(self):

return self.user.user.username

**Explanation**

This code defines the data structure for a healthcare application using Django's models. It creates a Patient model that stores basic details like contact, address, date of birth, and an image linked to a user account. The Doctor model is similar but also includes a status field, a category, and the date they joined the organization. An Admin\_Health\_CSV model is used to store CSV files containing health data along with a name for easy identification. The Search\_Data model records the results of health predictions, such as accuracy, prediction result, and the input values used, while linking the record to a specific patient. Feedback from users is collected in a Feedback model, which saves messages and the date they were sent. Each model has a string method that returns the username of the associated user for clarity when displaying records. Relationships between patients, doctors, and user accounts are established through foreign keys. This structure helps organize data in a clear and manageable way for the healthcare system. Overall, the models work together to provide a solid foundation for managing users, doctors, patients, health data, and feedback.

Login page

{% extends 'index.html' %}

{% load static %}

{% block body %}

{% if error == "pat" %}

<script>

alert('logged in successfully');

window.location = "{% url 'admin\_home' %}";

</script>

{% endif %}

{% if error == "notmember" %}

<script>

alert('Your information is pending. Please update your profile first, then continue this service.');

window.location = "{% url 'service\_home' %}";

</script>

{% endif %}

{% if error == "pat2" %}

<script>

alert('logged in successfully');

window.location = "{% url 'admin\_home' %}";

</script>

{% endif %}

{% if error == "not" %}

<script>

alert('Username & Password are not Matching');

</script>

{% endif %}

<section class="logins py-5">

<div class="container py-xl-5 py-lg-3">

<div class="title-section mb-md-5 mb-4">

<h6 class="w3ls-title-sub"></h6>

<br><br>

<center>

<h3 class="w3ls-title text-uppercase text-dark font-weight-bold">Login</h3>

</center>

</div>

<hr/>

<div class="login px-sm-4 mx-auto mw-100 login-wrapper">

<form class="login-wrapper" action="" method="post">

{% csrf\_token %}

<div class="form-group">

<label>Username</label>

<input type="text" class="form-control" name="uname" placeholder="Enter Username" required="">

<small id="emailHelp" class="form-text text-muted">Your privacy is our priority.</small>

</div>

<div class="form-group">

<label>Password</label>

<input type="password" class="form-control" name="pwd" placeholder="Enter Your Password" required="">

</div>

<center>

<button type="submit" class="btn submit mt-4" style="background: #6a6b6c;">Login</button>

</center>

</form>

</div>

</div>

</section>

{% endblock %}

**Explanation**

This template is designed to render a login page for the web application in a clear and user-friendly way. It starts by extending a base template to maintain a consistent look and feel across the website. The template also loads static resources, such as CSS and JavaScript files, to ensure proper styling and functionality. It uses conditional checks to determine if any error messages need to be displayed to the user. For instance, if the error variable equals "pat" or "pat2", a success alert is shown, and the user is redirected to the admin home page. If the error is "notmember", the user is informed that their profile update is required before accessing further services. Similarly, if the error is "not", an alert notifies the user that the provided username and password do not match. The page layout includes a prominent "Login" heading centered on the screen, making it immediately clear what action is expected. A well-structured form is presented where users can enter their username and password securely, with clear labels and placeholders guiding the input process. Finally, a submit button is provided in the center of the form to allow users to log in, completing the overall simple and effective design of the login interface.

**Reset password**

{% extends 'index.html' %}

{% load static %}

{% block body %}

<!-- register -->

{% if terror == "yes" %}

<script>

alert('Password Changed.....');

window.location = "{% url 'logout' %}";

</script>

{% endif %}

{% if terror == "not" %}

<script>

alert('New Password and Confirm Password are not match');

</script>

{% endif %}

<section class="logins py-5">

<div class="container py-xl-5 py-lg-3">

<div class="title-section mb-md-5 mb-4">

<h6 class="w3ls-title-sub"></h6>

<br>

<h3 class="w3ls-title text-uppercase text-dark font-weight-bold">Reset Password</h3>

</div>

<hr/>

<div class="login px-sm-12" style="width:100%">

<form action="" method="post" enctype="multipart/form-data">

{% csrf\_token %}

<div class="form-group row">

<div class="col-md-12">

<label>Old Password</label>

<input type="password" class="form-control" name="pwd3" required="">

</div>

</div>

<div class="form-group row">

<div class="col-md-12">

<label>New Password</label>

<input type="password" class="form-control" name="pwd1" required="">

</div>

</div>

<div class="form-group row">

<div class="col-md-12">

<label>Confirm Password</label>

<input type="password" class="form-control" name="pwd2" required="">

</div>

</div>

<button type="submit" class="btn submit mt-4" style="background-color: #6a6b6c;">Reset Password</button>

</form>

</div>

</div>

</section>

<!-- //register -->

**Explanation**

This template extends a base page to maintain a consistent design throughout the website. It loads necessary static files, such as stylesheets and scripts, ensuring that the page looks good and works properly. The template checks for a variable named "terror" to determine if any alerts should be shown to the user. If "terror" equals "yes", a JavaScript alert informs the user that their password has been changed and then redirects them to the logout page. If "terror" equals "not", an alert warns that the new password and confirmation password do not match. The main section of the page displays a "Reset Password" form that is easy to understand. It includes input fields for the old password, the new password, and a confirmation of the new password. Each field is clearly labeled so that users know what information to enter. A CSRF token is included in the form to protect against cross-site request forgery attacks. Finally, when the form is submitted, the new password details are sent securely to the server for processing.

Predict and analysis page

{% extends 'index.html' %}

{% load static %}

{% block body %}

<section class="logins py-5">

<div class="container py-xl-5 py-lg-3">

<div class="title-section mb-md-5 mb-4">

<h6 class="w3ls-title-sub" style="color:red"></h6><br><br><center>

<h3 class="w3ls-title text-uppercase text-dark font-weight-bold">Heart Prediction Output</h3></center>

</div><hr/>

<div class="login px-sm-12" style="width:100%">

<h1 align='center' style='color:#414141'>Prediction output</h1><hr><center><p><b>Accuracy (%) is :</b> {{ accuracy }} </p><p><b>Result:</b>

{% if pred == "0" %}

<span style='color:green'><strong>You are healthy.</strong></span>

{% else %}

<span style='color:red'><strong>You may posses a risk of heart disease.</strong></span>

{% endif %}

</p></center>

</div>

{% if pred != "0" %}

<div class="container-fluid" style="width:90%;margin-top:3%">

<div class="container-fluid">

<h1 align="center" class="w3ls-title text-uppercase text-dark font-weight-bold">Contact our doctors</h1>

</div><hr>

<table id="example" class="display" style="width:100%">

<thead>

<tr>

<th>#</th>

<th>Image</th>

<th>Full Name</th>

<th>Email</th>

<th>Contact</th>

<th>Address</th>

</tr>

</thead>

<tbody>

{% for i in doctor %}

<tr>

<td>{{forloop.counter}}</td>

<td><img src="{{i.image.url}}" style="width:80px;height:90px"></td>

<td>{{i.user.first\_name}} {{i.user.last\_name}}</td>

<td>{{i.user.email}}</td>

<td>{{i.contact}}</td>

<td>{{i.address}}</td>

</tr>

{% empty %}

<tr>

<td colspan="5">No Record Found.</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

{% endif %}

</div>

</section>

{% endblock %}

**Explanation**

This template builds on a base layout to keep the website's look consistent. It loads static files to ensure that styles and scripts work correctly. Inside the main body, it sets up a section specifically for displaying heart prediction results. A prominent title, "Heart Prediction Output," is centered at the top of the page. The page then shows the prediction details, including the accuracy percentage of the analysis. A conditional check determines the result: if the prediction value is "0," it displays a green message saying, "You are healthy." Otherwise, it shows a red warning indicating that there may be a risk of heart disease. If the prediction suggests risk, an additional section appears that lists doctors for consultation. This section presents a table containing doctors' images, names, emails, contact numbers, and addresses. Overall, the template clearly communicates the prediction results and offers help by connecting users to local doctors if needed.

**Registration page**

{% extends 'index.html' %}

{% load static %}

{% block body %}

<!-- register -->

{% if error == "create" %}

<script>

alert('Registration Successfull');

window.location="{% url 'login' %}";

</script>

{% endif %}

<section class="logins py-5">

<div class="container py-xl-5 py-lg-3">

<div class="title-section mb-md-5 mb-4">

<h6 class="w3ls-title-sub"></h6>

<h3 class="w3ls-title text-uppercase text-dark font-weight-bold">Register Now</h3>

</div>

<hr/>

<div class="login px-sm-12" style="width:100%">

<form action="" method="post" enctype="multipart/form-data">

{% csrf\_token %}

<div class="form-group row">

<div class="col-md-6">

<label>First Name</label>

<input type="text" class="form-control" name="fname" placeholder="First Name" required="">

</div>

<div class="col-md-6">

<label>Last Name</label>

<input type="text" class="form-control" name="lname" placeholder="Last Name" required="">

</div>

</div>

<div class="form-group row">

<div class="col-md-6">

<label>Username</label>

<input type="text" class="form-control" name="uname" placeholder="Username" required="">

</div>

<div class="col-md-6">

<label>Password</label>

<input type="password" class="form-control" name="pwd" placeholder="Password" required="">

</div>

</div>

<div class="form-group row">

<div class="col-md-6">

<label>Email</label>

<input type="email" class="form-control" name="email" placeholder="Enter Email" required="">

</div>

<div class="col-md-6">

<label>Contact</label>

<input type="text" class="form-control" name="contact" placeholder="Enter Contact" required="">

</div>

</div>

<div class="form-group row">

<div class="col-md-6">

<label>Date Of Birth</label>

<input type="date" class="form-control" name="dob" required="">

</div>

<div class="col-md-6">

<label>Image</label>

<input type="file" class="form-control" name="image" required="">

</div>

</div>

<div class="form-group row">

<div class="col-md-6">

<label class="mb-2">City</label>

<input type="text" class="form-control" name="add" id="password1" placeholder="Enter Address" required="">

</div>

<div class="col-md-6">

<label>User Type</label>

<div class="form-control">

Patient <input type="radio" name="type" style="margin-right:4%" required value="Patient">

Doctor <input type="radio" name="type" required value="Doctor">

</div>

</div>

</div>

<button type="submit" class="btn submit mt-4">Register</button>

<p class="text-center mt-5">

<a href="#">By clicking Register, I agree to your terms</a>

</p>

</form>

</div>

</div>

</section>

<!-- //register -->

{% endblock %}

**Explanation**

This registration page template extends a base layout to ensure consistency across the website. It includes a script that triggers an alert when a user successfully registers and then redirects them to the login page. The registration form is structured into different sections, collecting essential user details such as first and last names, username, password, email, and contact information. Additionally, users must provide their date of birth and upload a profile image.

The form also requires an address, labeled as "City," and asks the user to select their role as either a "Patient" or a "Doctor" using radio buttons. At the bottom, there is a submit button to complete the registration, along with a disclaimer stating that by registering, the user agrees to the website’s terms. This page is designed to be user-friendly, ensuring a smooth registration process while capturing all necessary information.