**SAMPLE CODE**

**Admin Views :**

from django.shortcuts import render, HttpResponse

from django.contrib import messages

from users.models import UserRegistrationModel

# Create your views here.

def AdminLoginCheck(request):

if request.method == 'POST':

usrid = request.POST.get('loginid')

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

print("User ID is = ", usrid)

if usrid == 'admin' and pswd == 'admin':

return render(request, 'admins/AdminHome.html')

else:

messages.success(request, 'Please Check Your Login Details')

return render(request, 'AdminLogin.html', {})

def AdminHome(request):

return render(request, 'admins/AdminHome.html')

def RegisterUsersView(request):

data = UserRegistrationModel.objects.all()

return render(request, 'admins/viewregisterusers.html', {'data': data})

def ActivaUsers(request):

if request.method == 'GET':

id = request.GET.get('uid')

status = 'activated'

print("PID = ", id, status)

UserRegistrationModel.objects.filter(id=id).update(status=status)

data = UserRegistrationModel.objects.all()

return render(request, 'admins/viewregisterusers.html', {'data': data})

def DeleteUsers(request):

if request.method == 'GET':

id = request.GET.get('uid')

print("Delete PID = ", id)

UserRegistrationModel.objects.filter(id=id).delete()

data = UserRegistrationModel.objects.all()

return render(request, 'admins/viewregisterusers.html', {'data': data})

def BlockUsers(request):

if request.method == 'GET':

id = request.GET.get('uid')

status = 'waiting'

print("PID = ", id, status)

UserRegistrationModel.objects.filter(id=id).update(status=status)

data = UserRegistrationModel.objects.all()

return render(request, 'admins/viewregisterusers.html', {'data': data})

def data\_v(request):

from django.conf import settings

import pandas as pd

path = settings.MEDIA\_ROOT + "//" + 'Training.csv'

d = pd.read\_csv(path)

# Drop the last column

if not d.empty:

d = d.iloc[:, :-1]

# d = d.head(50)

print(d)

return render(request,'admins/dataset.html', {'d': d})

**Base.html :**

{% load static %}

<!DOCTYPE html>

<html>

<head>

<!-- Basic -->

<meta charset="utf-8"/>

<meta http-equiv="X-UA-Compatible" content="IE=edge"/>

<!-- Mobile Metas -->

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"/>

<!-- Site Metas -->

<meta name="keywords" content=""/>

<meta name="description" content=""/>

<meta name="author" content=""/>

<title>Human Disease Prediction based on Symptoms</title>

<!-- slider stylesheet -->

<link rel="stylesheet" type="text/css"

href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.1.3/assets/owl.carousel.min.css"/>

<!-- bootstrap core css -->

<link rel="stylesheet" type="text/css" href="{%static 'css/bootstrap.css'%}"/>

<!-- fonts style -->

<link href="https://fonts.googleapis.com/css?family=Poppins:400,600,700&display=swap" rel="stylesheet">

<!-- Custom styles for this template -->

<link href="{%static 'css/style.css'%}" rel="stylesheet"/>

<!-- responsive style -->

<link href="{%static 'css/responsive.css'%}" rel="stylesheet"/>

<style>

body {

background-image: url("{% static 'images/2.jpg' %}");

background-repeat: no-repeat;

background-size: cover;

}

</style>

</head>

<body class="sub\_page">

<div class="hero\_area" style=" background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);;">

<!-- header section strats -->

<header class="header\_section">

<div class="container">

<nav class="navbar navbar-expand-lg custom\_nav-container ">

<a class="navbar-brand" href="{% url 'index' %}">

<!-- <img src="{%static 'images/logo.png'%}" alt=""> -->

<span style="color: whitesmoke;">

Human Disease Prediction based on Symptoms

</span>

</a>

<button class="navbar-toggler" type="button" data-toggle="collapse"

data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="s-1"> </span>

<span class="s-2"> </span>

<span class="s-3"> </span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<div class="d-flex ml-auto flex-column flex-lg-row align-items-center">

<ul class="navbar-nav ">

<style>

/\* Custom styles for the navigation menu \*/

.navbar {

background-color: #34495e; /\* Dark blue-gray background color for the navbar \*/

}

.navbar-brand img {

width: 50px; /\* Set the width of the logo \*/

height: auto; /\* Maintain the aspect ratio \*/

}

.navbar-toggler {

background-color: #2c3e50; /\* Dark blue-gray color for the toggle button \*/

}

.navbar-toggler span {

background-color: #ecf0f1; /\* Light gray color for the toggle button lines \*/

}

.navbar-nav .nav-item {

margin-right: 10px; /\* Add margin between navigation items \*/

}

.navbar-nav .nav-link {

color: #ecf0f1; /\* Light gray color for navigation links \*/

padding: 10px; /\* Increase padding for better spacing \*/

border-radius: 5px; /\* Add rounded corners to the links \*/

transition: background-color 0.3s; /\* Smooth transition for background color \*/

}

.navbar-nav .nav-link:hover {

background-color: #2c3e50; /\* Darker blue-gray background color on hover \*/

}

/\* Custom styles for specific navigation items \*/

.nav-link-home {

background-color: #3498db; /\* Blue background color for Home \*/

border-color: #3498db; /\* Border color same as the background color \*/

}

.nav-link-user {

background-color: #2ecc71; /\* Green background color for User \*/

border-color: #2ecc71; /\* Border color same as the background color \*/

}

.nav-link-admin {

background-color: #e74c3c; /\* Red background color for Admin \*/

border-color: #e74c3c; /\* Border color same as the background color \*/

}

.nav-link-register {

background-color: #f39c12; /\* Orange background color for Register \*/

border-color: #f39c12; /\* Border color same as the background color \*/

}

</style>

<!-- Your navigation menu with colored buttons and borders -->

<ul class="nav">

<li class="nav-item">

<a class="nav-link nav-link-home" href="{% url 'index' %}">Home</a>

</li>

<li class="nav-item">

<a class="nav-link nav-link-user" href="{% url 'UserLogin' %}">User</a>

</li>

<li class="nav-item">

<a class="nav-link nav-link-admin" href="{% url 'AdminLogin' %}">Admin</a>

</li>

<li class="nav-item">

<a class="nav-link nav-link-register" href="{% url 'UserRegister' %}">Register</a>

</li>

</ul>

</ul>

</div>

</div>

</nav>

</div>

</header>

<!-- end header section -->

</div>

{%block contents%}

{%endblock%}

<!-- footer section -->

<footer class="container-fluid footer\_section">

<div class="container">

<div class="row">

<div class="col-lg-7 col-md-9 mx-auto">

</div>

</div>

</div>

</footer>

<!-- footer section -->

<script src="{%static 'js/jquery-3.4.1.min.js'%}"></script>

<script src="{%static 'js/bootstrap.js'%}"></script>

</body>

</html>

**User views**  :

# Create your views here.

from django.shortcuts import render, HttpResponse

from django.contrib import messages

from .forms import UserRegistrationForm

from .models import UserRegistrationModel

from django.conf import settings

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

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

from sklearn.metrics import accuracy\_score

import numpy as np

import string

import random

import os

# Create your views here.

def UserRegisterActions(request):

if request.method == 'POST':

form = UserRegistrationForm(request.POST)

if form.is\_valid():

print('Data is Valid')

form.save()

messages.success(request, 'You have been successfully registered')

form = UserRegistrationForm()

return render(request, 'UserRegistrations.html', {'form': form})

else:

messages.success(request, 'Email or Mobile Already Existed')

print("Invalid form")

else:

form = UserRegistrationForm()

return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):

if request.method == "POST":

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

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

print("Login ID = ", loginid, ' Password = ', pswd)

try:

check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)

status = check.status

print('Status is = ', status)

if status == "activated":

request.session['id'] = check.id

request.session['loggeduser'] = check.name

request.session['loginid'] = loginid

request.session['email'] = check.email

print("User id At", check.id, status)

return render(request, 'users/UserHomePage.html', {})

else:

messages.success(request, 'Your Account Not at activated')

return render(request, 'UserLogin.html')

except Exception as e:

print('Exception is ', str(e))

pass

messages.success(request, 'Invalid Login id and password')

return render(request, 'UserLogin.html', {})

def UserHome(request):

return render(request, 'users/UserHomePage.html', {})

import pandas

path\_data=settings.MEDIA\_ROOT + "//" + 'Training.csv'

data=pandas.read\_csv(path\_data)

from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

data['prognosis']=lb.fit\_transform(data['prognosis'])

x=data.iloc[:,0:-1]

print(x.columns)

y=data.iloc[:,-1]

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

from sklearn.metrics import confusion\_matrix,accuracy\_score,precision\_score, recall\_score, f1\_score

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3)

import numpy as np

from sklearn.metrics import ConfusionMatrixDisplay

def algorithms(request):

# =============================================Naive Beyas=======================================

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(x\_train, y\_train)

y\_ = classifier.predict(x\_test)

a=accuracy\_score(y\_,y\_test)

b=precision\_score(y\_,y\_test, average='weighted')

c=recall\_score(y\_,y\_test, average='weighted')

d=f1\_score(y\_,y\_test, average='weighted')

print('Cofusion matrix for Naive Beyas')

print('accuracy score for is',a,b,c,d)

cm=confusion\_matrix(y\_,y\_test)

print(cm)

import matplotlib.pyplot as plt

confusion=ConfusionMatrixDisplay(confusion\_matrix=cm)

confusion.plot()

plt.title('Cofusion matrix for Naive Beyas')

plt.show()

# =============================================RandomForestClassifier=======================================

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()

rf.fit(x\_train, y\_train)

y= rf.predict(x\_test)

e=accuracy\_score(y,y\_test)

f=precision\_score(y,y\_test, average='weighted')

g=recall\_score(y,y\_test, average='weighted')

h=f1\_score(y,y\_test, average='weighted')

print('Cofusion matrix for RandomForestClassifier')

print('accuracy score is',e,f,g,h)

cm=confusion\_matrix(y,y\_test)

print(cm)

import matplotlib.pyplot as plt

confusion=ConfusionMatrixDisplay(confusion\_matrix=cm)

confusion.plot()

plt.title('Cofusion matrix for RandomForestClassifier')

plt.show()

return render(request,'users/accuracy.html',{'a':a,'b':b,'c':c,'d':d,'e':e,'f':f,'g':g,'h':h})

**GUI** :

root = Tk()

root.title(" Disease Prediction From Symptoms")

root.configure()

#these below are Symptom variables

Symptom1 = StringVar()

Symptom1.set(None)

Symptom2 = StringVar()

Symptom2.set(None)

Symptom3 = StringVar()

Symptom3.set(None)

Symptom4 = StringVar()

Symptom4.set(None)

Symptom5 = StringVar()

Symptom5.set(None)

w2 = Label(root, justify=LEFT, text=" Disease Prediction From Symptoms ")

w2.config(font=("Elephant", 30))

w2.grid(row=1, column=0, columnspan=2, padx=100)

NameLb1 = Label(root, text="")

NameLb1.config(font=("Elephant", 20))

NameLb1.grid(row=5, column=1, pady=10, sticky=W)

S1Lb = Label(root, text="Symptom 1")

S1Lb.config(font=("Elephant", 15))

S1Lb.grid(row=7, column=1, pady=10 , sticky=W)

S2Lb = Label(root, text="Symptom 2")

S2Lb.config(font=("Elephant", 15))

S2Lb.grid(row=8, column=1, pady=10, sticky=W)

S3Lb = Label(root, text="Symptom 3")

S3Lb.config(font=("Elephant", 15))

S3Lb.grid(row=9, column=1, pady=10, sticky=W)

S4Lb = Label(root, text="Symptom 4")

S4Lb.config(font=("Elephant", 15))

S4Lb.grid(row=10, column=1, pady=10, sticky=W)

S5Lb = Label(root, text="Symptom 5")

S5Lb.config(font=("Elephant", 15))

S5Lb.grid(row=11, column=1, pady=10, sticky=W)

lr = Button(root, text="Predict",height=2, width=20, command=message)

lr.config(font=("Elephant", 15))

lr.grid(row=15, column=1,pady=20)

OPTIONS = sorted(l1)

S1En = OptionMenu(root, Symptom1,\*OPTIONS)

S1En.grid(row=7, column=2)

S2En = OptionMenu(root, Symptom2,\*OPTIONS)

S2En.grid(row=8, column=2)

S3En = OptionMenu(root, Symptom3,\*OPTIONS)

S3En.grid(row=9, column=2)

S4En = OptionMenu(root, Symptom4,\*OPTIONS)

S4En.grid(row=10, column=2)

S5En = OptionMenu(root, Symptom5,\*OPTIONS)

S5En.grid(row=11, column=2)

NameLb = Label(root, text="")

NameLb.config(font=("Elephant", 20))

NameLb.grid(row=13, column=1, pady=10, sticky=W)

NameLb = Label(root, text="")

NameLb.config(font=("Elephant", 15))

NameLb.grid(row=18, column=1, pady=10, sticky=W)

t3 = Text(root, height=2, width=30)

t3.config(font=("Elephant", 20))

t3.grid(row=20, column=1 , padx=10)

root.mainloop()