**SAMPLE CODE**

**Urls.py**

"""recommendersystem URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/2.2/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.conf.urls import url

from django.conf.urls.static import static

from django.contrib import admin

from django.urls import path

from recommendersystem import settings

from simple.views import index, user, userregistration, adminlogin, adminloginaction, adminhome, adminbase, \

viewadminuserpage, activateusers, logout, uploadfile, storecsvdata, faslogin, fasloginaction, fashome,faspredection

from user.views import userlogincheck, userhome, userbase, userfitness, status, viewuserdailystatus,UserPredections

urlpatterns = [

url(r'^admin/', admin.site.urls),

url(r'^$', index, name="index"),

url(r'^index/', index, name="index"),

url(r'^user/',user,name="user"),

url(r'^userregistration/',userregistration,name="userregistration"),

url(r'^userlogincheck/', userlogincheck, name="userlogincheck"),

url(r'^userhome/', userhome, name="userhome"),

url(r'^userbase/', userbase, name="userbase"),

url(r'^uploadfile/',uploadfile,name="uploadfile"),

url(r'^storecsvdata/',storecsvdata,name="storecsvdata"),

url(r'^UserPredections/',UserPredections,name='UserPredections'),

url(r'^adminlogin/', adminlogin, name="adminlogin"),

url(r'^adminhome/', adminhome, name="adminhome"),

url(r'^adminbase/',adminbase,name="adminbase"),

url(r'^adminloginaction/', adminloginaction, name="adminloginaction"),

url(r'^viewadminuserpage/',viewadminuserpage, name="viewadminuserpage"),

url(r'^activateusers/$', activateusers, name="activateusers"),

url(r'^faslogin/', faslogin, name="faslogin"),

url(r'^fasloginaction/', fasloginaction, name="fasloginaction"),

url(r'^fashome/', fashome, name="fashome"),

url(r'^userfitness/', userfitness, name="userfitness"),

url(r'^status/', status, name="status"),

url(r'^viewuserdailystatus/',viewuserdailystatus,name="viewuserdailystatus"),

url(r'^faspredection/',faspredection,name='faspredection'),

url(r'^logout/',logout,name="logout"),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**simple.forms.py**

from django import forms

from django.core import validators

import datetime

from django.utils import timezone

from simple.models import userregistrationmodel, upload, dailystatus

class userregistrationform(forms.ModelForm):

name = forms.CharField(widget=forms.TextInput(), required=True, max\_length=100)

loginid = forms.CharField(widget=forms.TextInput(), required=True, max\_length=100)

password = forms.CharField(widget=forms.PasswordInput(), required=True, max\_length=100)

gender = forms.ChoiceField(choices=[('female','female'),('male','male')])

age = forms.CharField(widget=forms.NumberInput(), required=True, max\_length=100)

height = forms.FloatField(required=False, max\_value=250, min\_value=0, widget=forms.NumberInput(attrs={'id': 'form\_homework', 'step': "0.01"}))

weight = forms.CharField(widget=forms.NumberInput(), required=True, max\_length=100)

email = forms.EmailField(widget=forms.TextInput(), required=True)

contact = forms.CharField(widget=forms.NumberInput(), required=True, max\_length=100,validators=[validators.MaxLengthValidator(10),validators.MinLengthValidator(10)])

authkey = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max\_length=100)

status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max\_length=100)

class Meta:

model = userregistrationmodel

fields = ['name','loginid','password','gender','age','height','weight','email','contact','authkey','status']

class UploadfileForm(forms.ModelForm):

class Meta:

model = upload

fields = ('filename','description','file')

class dailystatusform(forms.ModelForm):

name = forms.CharField(widget=forms.TextInput(), required=True, max\_length=100)

excercise = forms.CharField(widget=forms.TextInput(),required=True,max\_length=100)

email = forms.CharField(widget=forms.TextInput(),required=True,max\_length=100)

contact = forms.CharField(widget=forms.NumberInput(),required=True,max\_length=100)

date = forms.DateField(widget=forms.DateInput())

#duration = forms.ChoiceField(choices=[('1hour','1hour'),('2hours','2hours'),('3hours','3hours'),('4hours','4hours'),('5hours','5hours'),('6hours','6hours'),('7hours','7hours'),('8hours','8hours'),('9hours','9hours'),('10hours','10hours'),('11hours','11hours'),('12hours','12hours'),('13hours','13hours'),('14hours','14hours'),('15hours','15hours'),('16hours','16hours'),('17hours','17hours'),('18hours','18hours'),('19hours','19hours'),('20hours','20hours')])

duration = forms.TimeField(widget=forms.TimeInput(format='%H:%M'),required=True)

caloriesburned = forms.CharField(widget=forms.NumberInput(),required=True,max\_length=100)

class Meta:

model = dailystatus

fields = ('name','excercise','email','contact','date','duration','caloriesburned')

**simple.models.py**

from django.db import models

class userregistrationmodel(models.Model):

name = models.CharField(max\_length=100)

loginid = models.CharField(max\_length=100)

password = models.CharField(max\_length=100)

gender = models.CharField(max\_length=100)

age = models.CharField(max\_length=100)

height = models.FloatField()

weight = models.CharField(max\_length=100)

email = models.EmailField()

contact = models.CharField(max\_length=100)

authkey = models.CharField(max\_length=100)

status = models.CharField(max\_length=100)

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

return self.email

class upload(models.Model):

#uuid = models.CharField(max\_length=30)

filename = models.CharField(max\_length=100)

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

file = models.FileField(upload\_to='files/pdfs/')

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

return self.filename

def delete(self, \*args, \*\*kwargs):

self.file.delete()

super().delete(\*args,\*\*kwargs)

class excercisesdata(models.Model):

age = models.CharField(max\_length=500)

weight = models.CharField(max\_length=300)

excercise1 = models.CharField(max\_length=300)

excercise2 = models.CharField(max\_length=300)

diet = models.CharField(max\_length=255)

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

return self.excercise1

class dailystatus(models.Model):

name = models.CharField(max\_length=100)

excercise = models.CharField(max\_length=100)

email = models.CharField(max\_length=100)

contact = models.FloatField(max\_length=100)

date = models.DateField(max\_length=50)

duration = models.CharField(max\_length=50)

caloriesburned = models.CharField(max\_length=100)

class Meta:

unique\_together = ('name', 'date',)

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

return self.email

**Simple.views.py**

import csv

from collections import defaultdict

from io import TextIOWrapper

from random import randint

from django.contrib import messages

from django.http import HttpResponseRedirect, HttpResponse

from django.shortcuts import render, redirect

from simple.forms import userregistrationform, UploadfileForm

from simple.models import userregistrationmodel, excercisesdata

from user.algorithms.anntest import NeuralNetwork

import numpy as np

import random

from user.algorithms.LogisticRegressionTest import runLogisticAlgo

def index(request):

return render(request,"index.html")

def userregistration(request):

if request.method == 'POST':

form = userregistrationform(request.POST)

if form.is\_valid():

form.save()

messages.success(request, 'you are successfully registred')

return HttpResponseRedirect('user')

else:

print('Invalid')

else:

form = userregistrationform()

return render(request,"user/userregistration.html",{'form':form})

def user(request):

context={}

return render(request,'user/user.html',context)

def adminlogin(request):

return render(request,"admin/adminlogin.html")

def adminloginaction(request):

if request.method == "POST":

#if request.method == "POST":

usid = request.POST.get('username')

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

if usid == 'admin' and pswd == 'admin':

return render(request,'admin/adminhome.html')

else:

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

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

return render(request,'adminlogin.html')

def adminbase(request):

return render(request,"adminbase.html")

def adminhome(request):

return render(request,'admin/adminhome.html')

def viewadminuserpage(request):

userdata = userregistrationmodel.objects.all()

return render(request,'admin/viewuserdata.html',{'object':userdata})

def activateusers(request):

if request.method == 'GET':

usid = request.GET.get('pid')

authkey = random\_with\_N\_digits(8)

status = 'activated'

print("USID = ",usid,authkey,status)

userregistrationmodel.objects.filter(id=usid).update(authkey=authkey , status=status)

userdata = userregistrationmodel.objects.all()

return render(request,'admin/viewuserdata.html',{'object':userdata})

def random\_with\_N\_digits(n):

range\_start = 10\*\*(n-1)

range\_end = (10\*\*n)-1

return randint(range\_start, range\_end)

def logout(request):

return render(request,'index.html')

def uploadfile(request):

if request.method == 'POST':

form = UploadfileForm(request.POST, request.FILES)

if form.is\_valid():

form.save()

return redirect('index.html')

else:

form = UploadfileForm()

return render(request, 'fas/uploadfile.html', {'form': form})

def storecsvdata(request):

if request.method == 'POST':

#age = request.POST.get('age')

#weight = request.POST.get('weight')

csvfile = TextIOWrapper(request.FILES['file'])

print(csvfile)

columns = defaultdict(list)

storecsvdata = csv.DictReader(csvfile)

# row1 = next(storecsvdata)

for row1 in storecsvdata:

age = row1["age"]

weight = row1["weight"]

excercise1 = row1["excercise1"]

excercise2 = row1["excercise2"]

diet = row1["diet"]

excercisesdata.objects.create(age=age, weight=weight, excercise1=excercise1,

excercise2=excercise2, diet=diet)

print("Name is ", csvfile)

return HttpResponse('CSV file successful uploaded')

else:

return render(request, 'fas/uploadfile.html', {})

def faslogin(request):

return render(request,"fas/faslogin.html")

def fasloginaction(request):

if request.method == "POST":

#if request.method == "POST":

usid = request.POST.get('username')

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

if usid == 'fas' and pswd == 'fas':

return render(request,'fas/fashome.html')

else:

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

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

return render(request,'fas/faslogin.html')

def fashome(request):

return render(request,"fas/fashome.html")

def faspredection(request):

dict = {}

listUsers = userregistrationmodel.objects.all()

dataLlist = []

idlist = []

for users in listUsers:

list = []

userid = users.loginid

v0 = int(users.id % 2)

if v0 == 0:

idlist.append(0)

else:

idlist.append(1)

# idlist.append(users.id)

# print('Registerd users = ',userid)

heightandweight = userregistrationmodel.objects.get(loginid=userid)

# print(heightandweight.age)

age = heightandweight.age

height = heightandweight.height

weight = heightandweight.weight

v1 = int(int(age) % 2)

print("V1 == ", v1)

if v1 == 0:

list.append(0)

else:

list.append(1)

# list.append(age)

v2 = int(int(height) % 2)

if v2 == 0:

list.append(0)

else:

list.append(1)

# list.append(height)

v3 = int(int(weight) % 2)

if v3 == 0:

list.append(0)

else:

list.append(1)

# list.append(weight)

dataLlist.append(list)

neural\_network = NeuralNetwork()

print("Beginning Randomly Generated Weights: ")

dict.update({"genweight": neural\_network.synaptic\_weights})

list = [[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]]

# training data consisting of 4 examples--3 input values and 1 output

print("Trainfin Dataset ", dataLlist)

print("Test Dataset ", idlist)

training\_inputs = np.array(dataLlist)

# training\_outputs = np.array([[0, 1, 1, 0]]).T

training\_outputs = np.array([idlist]).T

# training taking plac

neural\_network.train(training\_inputs, training\_outputs, 15000)

print("Ending Weights After Training: ")

dict.update({"aftrtraining": neural\_network.synaptic\_weights})

pathList = [0, 1]

user\_input\_one = random.choice(pathList) # str(input("User Input One: "))

user\_input\_two = random.choice(pathList) # str(input("User Input Two: "))

user\_input\_three = random.choice(pathList) # str(input("User Input Three: "))

dict.update({"newsiut": [user\_input\_one, user\_input\_two, user\_input\_three]})

print("New Output data: ")

dict.update({'result': neural\_network.think(np.array([user\_input\_one, user\_input\_two, user\_input\_three]))})

print("Its Done ")

lgdict = runLogisticAlgo()

return render(request, 'fas/faspredectiontest.html', {'dict': dict, 'lgdict': lgdict})

**users.views.py**

from django.contrib import messages

from django.shortcuts import render,HttpResponseRedirect,HttpResponse

from simple.forms import dailystatusform

from simple.models import userregistrationmodel, excercisesdata, dailystatus

from datetime import date

from .algorithms.anntest import NeuralNetwork

import numpy as np

import random

from .algorithms.LogisticRegressionTest import runLogisticAlgo

def userlogincheck(request):

if request.method == "POST":

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

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

try:

check = userregistrationmodel.objects.get(loginid=usid, password=pswd)

request.session['stuid'] = check.loginid

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

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

print(check.age)

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

status = check.status

print("stu id ",check.loginid)

if status == "activated":

#request.session['stuid'] = check.loginid

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

return render(request,'user/userhome.html')

else:

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

return render(request,'user/user.html')

return render(request,'user/userhome.html')

except Exception as e:

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

messages.success(request, 'Invalid Login Details')

return render(request,'user/user.html')

def userfitness(request):

ag=request.session['age']

wt=request.session['weight']

print('age = ',ag,' Weight = ',wt)

qs=excercisesdata.objects.filter(age=ag,weight=wt)

print(qs)

for x in qs:

print(x.diet)

return render(request,"user/userfitness.html",{"message":qs})

def userhome(request):

return render(request, 'user/userhome.html')

def userbase(request):

return render(request,"userbase.html")

def status(request):

if request.method == 'POST':

form = dailystatusform(request.POST)

if form.is\_valid():

form.save()

messages.success(request, 'you are successfully status updated')

return HttpResponseRedirect('user')

else:

print('Invalid')

else:

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

name=request.session['loggedstu']

today = date.today()

data ={'name':name,'date':today}

form = dailystatusform(data)

#form = dailystatusform()

return render(request,"user/status.html",{'form':form})

def viewuserdailystatus(request):

dict = dailystatus.objects.all()

return render(request,'admin/viewuserdailystatus.html',{'object':dict})

def UserPredections(request):

dict = {}

listUsers = userregistrationmodel.objects.all()

dataLlist = []

idlist = []

for users in listUsers:

list = []

userid = users.loginid

v0 = int(users.id%2)

if v0 == 0:

idlist.append(0)

else:

idlist.append(1)

#idlist.append(users.id)

#print('Registerd users = ',userid)

heightandweight = userregistrationmodel.objects.get(loginid=userid)

#print(heightandweight.age)

age = heightandweight.age

height = heightandweight.height

weight = heightandweight.weight

v1 = int(int(age)%2)

print("V1 == ",v1)

if v1 == 0:

list.append(0)

else:

list.append(1)

#list.append(age)

v2= int(int(height)%2)

if v2==0:

list.append(0)

else:

list.append(1)

#list.append(height)

v3 = int(int(weight)%2)

if v3==0:

list.append(0)

else:

list.append(1)

#list.append(weight)

dataLlist.append(list)

neural\_network = NeuralNetwork()

print("Beginning Randomly Generated Weights: ")

dict.update({"genweight":neural\_network.synaptic\_weights})

list = [[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]]

# training data consisting of 4 examples--3 input values and 1 output

print("Trainfin Dataset ",dataLlist)

print("Test Dataset ",idlist)

training\_inputs = np.array(dataLlist)

#training\_outputs = np.array([[0, 1, 1, 0]]).T

training\_outputs = np.array([idlist]).T

# training taking plac

neural\_network.train(training\_inputs, training\_outputs, 15000)

print("Ending Weights After Training: ")

dict.update({"aftrtraining":neural\_network.synaptic\_weights})

pathList = [0,1]

user\_input\_one = random.choice(pathList) #str(input("User Input One: "))

user\_input\_two = random.choice(pathList) #str(input("User Input Two: "))

user\_input\_three = random.choice(pathList) #str(input("User Input Three: "))

dict.update({"newsiut": [user\_input\_one, user\_input\_two, user\_input\_three]})

print("New Output data: ")

dict.update({'result':neural\_network.think(np.array([user\_input\_one, user\_input\_two, user\_input\_three]))})

print("Its Done ")

lgdict = runLogisticAlgo()

return render(request,'user/userpredections.html',{'dict':dict,'lgdict':lgdict})

**logisticregression.py**

#import pandas

import pandas as pd

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

# import the class

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

# import required modules

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

#%matplotlib inline

from django.conf import settings

def runLogisticAlgo():

dict = {}

col\_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']

# load dataset

filePath = settings.MEDIA\_ROOT+"\\"+"fitnestdiabitis.csv"

pima = pd.read\_csv(filePath, header=None, names=col\_names)

pima.head()

#split dataset in features and target variable

feature\_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']

X = pima[feature\_cols] # Features

y = pima.label # Target variable

# split X and y into training and testing sets

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.25,random\_state=0)

# instantiate the model (using the default parameters)

logreg = LogisticRegression()

# fit the model with data

logreg.fit(X\_train,y\_train)

y\_pred=logreg.predict(X\_test)

# import the metrics class

cnf\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)

cnf\_matrix

class\_names=[0,1] # name of classes

fig, ax = plt.subplots()

tick\_marks = np.arange(len(class\_names))

plt.xticks(tick\_marks, class\_names)

plt.yticks(tick\_marks, class\_names)

# create heatmap

sns.heatmap(pd.DataFrame(cnf\_matrix), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

plt.show()

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

print("Precision:",metrics.precision\_score(y\_test, y\_pred))

print("Recall:",metrics.recall\_score(y\_test, y\_pred))

dict.update({'Accuracy':metrics.accuracy\_score(y\_test, y\_pred)})

dict.update({'Precision': metrics.precision\_score(y\_test, y\_pred)})

dict.update({'Recall': metrics.recall\_score(y\_test, y\_pred)})

y\_pred\_proba = logreg.predict\_proba(X\_test)[::,1]

fpr, tpr, \_ = metrics.roc\_curve(y\_test, y\_pred\_proba)

auc = metrics.roc\_auc\_score(y\_test, y\_pred\_proba)

plt.plot(fpr,tpr,label="data 1, auc="+str(auc))

plt.legend(loc=4)

plt.show()

return dict

#if \_\_name\_\_=='\_\_main\_\_':

#runLogisticAlgo()

**Adminbase.html**

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<title>BodyFit - Free Bootstrap 4 Template by Colorlib</title>

<meta charset="utf-8">

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

<link href="https://fonts.googleapis.com/css?family=Poppins:300,400,500,600,700,800" rel="stylesheet">

<link rel="stylesheet" href="{% static 'css/open-iconic-bootstrap.min.css' %}">

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

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

<link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">

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

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

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

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

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

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

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

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

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-dark ftco\_navbar bg-dark ftco-navbar-light" id="ftco-navbar">

<div class="container">

<a class="navbar-brand" href="index.html">BodyFit<small>Fitness</small></a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#ftco-nav" aria-controls="ftco-nav" aria-expanded="false" aria-label="Toggle navigation">

<span class="oi oi-menu"></span> Menu

</button>

<div class="collapse navbar-collapse" id="ftco-nav">

<ul class="navbar-nav ml-auto">

<li class="nav-item active"><a href="{% url 'adminhome' %}"class="nav-link">Home</a></li>

<li class="nav-item"><a href="{% url 'viewadminuserpage' %}" class="nav-link">Userdetails</a></li>

<li class="nav-item"><a href="{% url 'viewuserdailystatus' %}" class="nav-link">Userdailystatus</a></li>

<li class="nav-item"><a href="{% url 'logout' %}"class="nav-link">LOGOUT</a></li>

<!--<li class="nav-item"><a href="blog.html" class="nav-link">Blog</a></li>

<li class="nav-item"><a href="contact.html" class="nav-link">Contact</a></li>-->

</ul>

</div>

</div>

</nav>

<!-- END nav -->

<section class="home-slider owl-carousel">

<div class="slider-item" style="background-image: url( '{% static 'images/bg\_1.jpg' %}');">

<div class="overlay"></div>

<div class="container">

<div class="row slider-text align-items-center" data-scrollax-parent="true">

<div class="col-md-5 col-sm-12 ftco-animate">

<h2 class="mb-4">Recommender System with Artificial Intelligence for Fitness

Assistance System</h2>

<!--<p class="mb-4 mb-md-5">A small river named Duden flows by their place and supplies it with the necessary regelialia.</p>-->

<!--<p><a href="#" class="btn btn-primary p-3 px-xl-4 py-xl-3">Get Started Now</a></p>-->

</div>

</div>

</div>

</div>

<div class="slider-item" style="background-image: url('{% static 'images/bg\_2.jpg' %}');">

<div class="overlay"></div>

<div class="container">

<div class="row slider-text align-items-center" data-scrollax-parent="true">

<div class="col-md-5 col-sm-12 ftco-animate">

<h2 class="mb-4">Recommender System with Artificial Intelligence for Fitness

Assistance System</h2>

<!--<p class="mb-4 mb-md-5">A small river named Duden flows by their place and supplies it with the necessary regelialia.</p>-->

<!--<p><a href="#" class="btn btn-primary p-3 px-xl-4 py-xl-3">Get Started Now</a></p>-->

</div>

</div>

</div>

</div>

</section>

{% block contents %}

{% endblock %}

<footer class="ftco-footer ftco-section img">

<center><p><!-- Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0. -->

Copyright &copy;<script>document.write(new Date().getFullYear());</script> All rights reserved | This template is made with <i class="icon-heart" aria-hidden="true"></i> by <a href="https://colorlib.com" target="\_blank">Colorlib</a>

<!-- Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0. --></p></center>

<!-- </div>

</div>

</div>-->

</footer>

<!-- loader -->

<div id="ftco-loader" class="show fullscreen"><svg class="circular" width="48px" height="48px"><circle class="path-bg" cx="24" cy="24" r="22" fill="none" stroke-width="4" stroke="#eeeeee"/><circle class="path" cx="24" cy="24" r="22" fill="none" stroke-width="4" stroke-miterlimit="10" stroke="#F96D00"/></svg></div>

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

<script src="{% static 'js/jquery-migrate-3.0.1.min.js' %}"></script>

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

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

<script src="{% static 'js/jquery.easing.1.3.js' %}"></script>

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

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

<script src="{% static 'js/owl.carousel.min.js' %}"></script>

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

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

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

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

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

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

<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBVWaKrjvy3MaE7SQ74\_uJiULgl1JY0H2s&sensor=false"></script>

<script src="{% static 'js/google-map.js' %}"></script>

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

</body>

</html>

**Userpredections.html**

{% extends 'userbase.html' %}

{% load static %}

{% block contents %}

<div>

<center><h2><b>User Predections Summery with ANN</b></h2></center>

<center>

<table border="2px solid blue" align="center">

<tr><td>Result</td> <td>{{dict.result}}</td></tr>

<tr><td>Situavations</td><td>{{dict.newsiut}}</td></tr>

<tr><td>After Training</td> <td style="font-weigt:bold;color:RED">{{dict.aftrtraining}}</td></tr>

<tr><td>Generated Weights</td> <td style="font-weigt:bold;color:RED">{{dict.genweight}}</td></tr>

<tr><td></td><td>Logistic Regression</td></tr>

<tr><td>Accuracy</td><td>{{lgdict.Accuracy}}</td></tr>

<tr><td>Precision</td><td>{{lgdict.Precision}}</td></tr>

<tr><td>Recall</td><td>{{lgdict.Recall}}</td></tr>

</table>

</center>

</div>

{% endblock %}

**Userregistration.html**

{% extends 'base.html' %}

{% block contents %}

<div>

<center><h2><b>User Registration Here</b></h2></center>

<p><form method="POST" action="{% url 'userregistration' %}">

{% csrf\_token %}

<center>

<table>

<tr><td>User Name</td><td>{{form.name}}</td></tr>

<tr><td>Login Name</td><td>{{form.loginid}}</td></tr>

<tr><td>Password</td><td>{{form.password}}</td></tr>

<tr><td>Gender</td><td>{{form.gender}}</td></tr>

<tr><td>Age</td><td>{{form.age}}</td></tr>

<tr><td>Height in (cm)</td><td>{{form.height}}</td></tr>

<tr><td>Weight</td><td>{{form.weight}}</td></tr>

<tr><td>Email</td><td>{{form.email}}</td></tr>

<tr><td>Contact</td><td>{{form.contact}}</td></tr>

<tr><td></td><td>{{form.authkey}}</td></tr>

<tr><td></td><td>{{form.status}}</td></tr>

<tr><td><button type="submit">Register</button></td></tr>

<center>

{% if messages %}

{% for message in messages %}

<font color='RED'>{{message}}</font>

{% endfor %}

{% endif %}

</center>

</table>

</center>

</form>

{% endblock %}

**Adminviewdailystatus.html**

{% extends 'adminbase.html' %}

{% load static %}

{% block contents %}

<div class="container">

<!-- Example row of columns -->

<h2><b>Admin View User Daily Status<b> </h2>

<table border="2px">

<tr>

<th>S.No</th>

<th >Name</th>

<th>Excercise</th>

<th>Email</th>

<th>Contact</th>

<th>Date</th>

<th>Duration</th>

<th>Calories Burned</th>

</tr>

{% for i in object %}

<tr>

<td style="color:brown">{{i.id}}</td>

<td style="color:darkolivegreen">{{i.name}}</td>

<td style="color:brown">{{i.excercise}}</td>

<td style="color:darkolivegreen">{{i.email}}</td>

<td style="color:brown">{{i.contact}}</td>

<td style="color:darkolivegreen">{{i.date}}</td>

<td style="color:brown">{{i.duration}}</td>

<td style="color:darkolivegreen">{{i.caloriesburned}}</td>

</tr>

{% endfor %}

</table>

{% endblock %}

**Userbase.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<title>BodyFit - Free Bootstrap 4 Template by Colorlib</title>

<meta charset="utf-8">

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

<link href="https://fonts.googleapis.com/css?family=Poppins:300,400,500,600,700,800" rel="stylesheet">

<link rel="stylesheet" href="{% static 'css/open-iconic-bootstrap.min.css' %}">

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

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

<link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">

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

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

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

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

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

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

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

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

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-dark ftco\_navbar bg-dark ftco-navbar-light" id="ftco-navbar">

<div class="container">

<a class="navbar-brand" href="index.html">BodyFit<small>Fitness</small></a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#ftco-nav" aria-controls="ftco-nav" aria-expanded="false" aria-label="Toggle navigation">

<span class="oi oi-menu"></span> Menu

</button>

<div class="collapse navbar-collapse" id="ftco-nav">

<ul class="navbar-nav ml-auto">

<li class="nav-item "><a href="{% url 'userhome' %}" class="nav-link">Home</a></li>

<li class="nav-item"><a href="{% url 'userfitness'%}" class="nav-link">ExerciseDetails</a></li>

<li class="nav-item"><a href="{% url 'status'%}" class="nav-link">Daily status</a></li>

<li class="nav-item"><a href="{% url 'UserPredections'%}" class="nav-link">User Predection</a></li>

<li class="nav-item"><a href="{% url 'logout' %}"class="nav-link">LOGOUT</a></li>

<!--<li class="nav-item"><a href="blog.html" class="nav-link">Blog</a></li>

<li class="nav-item"><a href="contact.html" class="nav-link">Contact</a></li>-->

</ul>

</div>

</div>

</nav>

<!-- END nav -->

<section class="home-slider owl-carousel">

<div class="slider-item" style="background-image: url( '{% static 'images/bg\_1.jpg' %}');">

<div class="overlay"></div>

<div class="container">

<div class="row slider-text align-items-center" data-scrollax-parent="true">

<div class="col-md-5 col-sm-12 ftco-animate">

<h2 class="mb-4">Recommender System with Artificial Intelligence for Fitness

Assistance System</h2>

<!--<p class="mb-4 mb-md-5">A small river named Duden flows by their place and supplies it with the necessary regelialia.</p>-->

<!--<p><a href="#" class="btn btn-primary p-3 px-xl-4 py-xl-3">Get Started Now</a></p>-->

</div>

</div>

</div>

</div>

<div class="slider-item" style="background-image: url('{% static 'images/bg\_2.jpg' %}');">

<div class="overlay"></div>

<div class="container">

<div class="row slider-text align-items-center" data-scrollax-parent="true">

<div class="col-md-5 col-sm-12 ftco-animate">

<h2 class="mb-4">Recommender System with Artificial Intelligence for Fitness

Assistance System</h2>

<!--<p class="mb-4 mb-md-5">A small river named Duden flows by their place and supplies it with the necessary regelialia.</p>-->

<!--<p><a href="#" class="btn btn-primary p-3 px-xl-4 py-xl-3">Get Started Now</a></p>-->

</div>

</div>

</div>

</div>

</section>

{% block contents %}

{% endblock %}

<footer class="ftco-footer ftco-section img">

<center><p><!-- Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0. -->

Copyright &copy;<script>document.write(new Date().getFullYear());</script> All rights reserved | This template is made with <i class="icon-heart" aria-hidden="true"></i> by <a href="https://colorlib.com" target="\_blank">Colorlib</a>

<!-- Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0. --></p></center>

<!-- </div>

</div>

</div>-->

</footer>

<!-- loader -->

<div id="ftco-loader" class="show fullscreen"><svg class="circular" width="48px" height="48px"><circle class="path-bg" cx="24" cy="24" r="22" fill="none" stroke-width="4" stroke="#eeeeee"/><circle class="path" cx="24" cy="24" r="22" fill="none" stroke-width="4" stroke-miterlimit="10" stroke="#F96D00"/></svg></div>

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

<script src="{% static 'js/jquery-migrate-3.0.1.min.js' %}"></script>

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

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

<script src="{% static 'js/jquery.easing.1.3.js' %}"></script>

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

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

<script src="{% static 'js/owl.carousel.min.js' %}"></script>

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

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

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

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

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

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

<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBVWaKrjvy3MaE7SQ74\_uJiULgl1JY0H2s&sensor=false"></script>

<script src="{% static 'js/google-map.js' %}"></script>

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

</body>

</html>

<title>Title</title>

</head>

<body>

</body>

</html>