**Pet Owners Communication Systems**

**models.py:**

from django.contrib.auth.models import User

from django.db import models

from django.utils import timezone

# Create your models here.

class Contact(models.Model):

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

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

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

message = models.TextField()

class Customer(models.Model):

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

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

mobile = models.BigIntegerField()

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

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

status = models.CharField(max\_length=100, default='Pending')

image=models.FileField()

class Notifications(models.Model):

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

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

ndate\_time = models.DateTimeField(auto\_now=True)

class Meta:

db\_table = "Notifications"

class Doctors(models.Model):

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

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

license\_number= models.CharField(max\_length=100)

contact\_number= models.BigIntegerField()

email= models.EmailField()

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

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

experience = models.IntegerField()

image=models.FileField()

status = models.CharField(max\_length=100, default='Pending')

class Pet(models.Model):

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

email=models.EmailField()

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

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

age=models.IntegerField()

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

weight=models.IntegerField()

height=models.FloatField()

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

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

image=models.ImageField()

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

sell\_for = models.CharField(max\_length=100,default='No')

# models.py

class Appoitments(models.Model):

doctor\_email = models.EmailField()

customer\_email = models.EmailField()

select\_pet = models.CharField(max\_length=100)

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

bookings\_date\_time=models.DateTimeField(auto\_now=True)

appoitments\_date\_time=models.DateTimeField()

status = models.CharField(max\_length=100, default='Pending')

class Precautions(models.Model):

email = models.EmailField()

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

description=models.TextField(max\_length=100)

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

date\_time=models.DateTimeField(auto\_now=True)

class Quote(models.Model):

doctor\_email = models.CharField(max\_length=100)

customer\_email = models.CharField(max\_length=30)

title=models.CharField(max\_length=40)

description=models.CharField(max\_length=20)

date\_time=models.DateTimeField(auto\_now=True)

class Prescriptions(models.Model):

appoitments = models.ForeignKey(Appoitments,on\_delete=models.CASCADE)

doctor\_email = models.EmailField()

customer\_email = models.EmailField()

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

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

prescription\_file = models.FileField()

date\_time=models.DateTimeField(auto\_now=True)

class Reviews(models.Model):

customer\_email = models.CharField(max\_length=100)

doctors = models.ForeignKey(Doctors,on\_delete=models.CASCADE)

reviews=models.TextField(max\_length=100)

ratings=models.IntegerField()

date\_time=models.DateTimeField(auto\_now=True)

class Admin(models.Model):

email = models.EmailField()

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

class Meta:

db\_table = "Admin"

class Friends\_Requests(models.Model):

customers = models.ForeignKey(Customer,on\_delete=models.CASCADE)

from\_email = models.EmailField()

to\_email = models.EmailField()

status = models.CharField(max\_length=100, default='Pending')

class Meta:

db\_table = "Friends\_Requests"

unique\_together = ('from\_email', 'to\_email')

class Add\_Posts(models.Model):

email = models.EmailField()

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

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

only\_for = models.CharField(max\_length=100)

pfile = models.FileField()

status = models.CharField(max\_length=100, default='Pending')

date\_time=models.DateTimeField(auto\_now=True)

class Likes(models.Model):

post = models.ForeignKey(Add\_Posts, on\_delete=models.CASCADE)

customer\_email = models.EmailField()

date\_time = models.DateTimeField(default=timezone.now)

class Meta:

unique\_together = ('post', 'customer\_email')

class Add\_Comments(models.Model):

posts = models.ForeignKey(Add\_Posts, on\_delete=models.CASCADE)

customer\_email = models.EmailField()

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

date\_time=models.DateTimeField(auto\_now=True)

class Interest(models.Model):

customer = models.ForeignKey(Customer, on\_delete=models.CASCADE)

pet = models.ForeignKey(Pet, on\_delete=models.CASCADE)

interest\_date = models.DateTimeField(auto\_now\_add=True)

class Pets\_Likes(models.Model):

pets = models.ForeignKey(Pet, on\_delete=models.CASCADE)

customer\_email = models.EmailField()

date\_time = models.DateTimeField(default=timezone.now)

class Meta:

unique\_together = ('pets', 'customer\_email')

**forms.py:**

from django import forms

from .models import \*

class ContactForm(forms.ModelForm):

class Meta:

model = Contact

fields = "\_\_all\_\_"

class CustomerForm(forms.ModelForm):

class Meta:

model = Customer

exclude=['status']

class DoctorsForm(forms.ModelForm):

class Meta:

model = Doctors

exclude = ["status"]

class Add\_notificationsForm(forms.ModelForm):

class Meta:

model = Notifications

fields = "\_\_all\_\_"

class PetForm(forms.ModelForm):

class Meta:

model = Pet

exclude = ["sell\_for"]

# forms.py

class AppoitmentsForm(forms.ModelForm):

class Meta:

model = Appoitments

exclude=['status']

class PrecautionsForm(forms.ModelForm):

class Meta:

model = Precautions

fields = "\_\_all\_\_"

class QuoteForm(forms.ModelForm):

class Meta:

model = Quote

fields = "\_\_all\_\_"

class PrescriptionsForm(forms.ModelForm):

class Meta:

model = Prescriptions

fields = "\_\_all\_\_"

class ReviewsForm(forms.ModelForm):

class Meta:

model = Reviews

fields = "\_\_all\_\_"

class Friends\_RequestsForms(forms.ModelForm):

class Meta:

model = Friends\_Requests

exclude = ["status"]

class Add\_PostsForms(forms.ModelForm):

class Meta:

model = Add\_Posts

exclude = ["status"]

class Add\_CommentsForm(forms.ModelForm):

class Meta:

model = Add\_Comments

fields = "\_\_all\_\_"

**views.py:**

from itertools import chain

from django.db.models import Q, Count

from django.shortcuts import render, redirect, get\_object\_or\_404

from .forms import \*

from .models import \*

from django.http import HttpResponse, JsonResponse

# Create your views here.

def index(request):

return render(request, "index.html", {})

def about(request):

return render(request, "about.html", {})

def blog(request):

return render(request, "blog.html", {})

def blog\_details(request):

return render(request, "blog\_details.html", {})

def contact(request):

if request.method == "POST":

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

print(form.errors)

if form.is\_valid():

form.save()

return render(request, "contact.html", {"msg": "Successfully Saved"})

return render(request, "contact.html", {})

return render(request, "contact.html", {})

def services(request):

return render(request, "services.html", {})

def customer\_home(request):

return render(request, "customer\_home.html", {})

def customer\_login(request):

if request.method == 'POST':

email = request.POST["email"]

password = request.POST["password"]

if Customer.objects.filter(email=email).exists():

user = Customer.objects.get(email=email)

if user is not None:

pwd = user.password

if password == pwd:

if user.status == "Accepted":

request.session["email"] = email

return render(request, 'customer\_home.html', {})

else:

return render(request, "customer\_login.html", {"msg": "Account Is On Hold"})

else:

return render(request, "customer\_login.html", {"msg": "Wrong Passwords"})

else:

return render(request, "customer\_login.html", {"msg": ""})

return render(request, "customer\_login.html", {"msg": "Email Not Registered"})

return render(request, "customer\_login.html", {})

def customer\_register(request):

if request.method == 'POST':

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

email = request.POST['email']

if form.is\_valid():

email=form.cleaned\_data["email"]

if Customer.objects.filter(email=email).exists():

return render(request, 'customer\_registration.html', {"msg": "This Email is Already Exists"})

else:

form.is\_valid()

form.save()

return render(request, 'customer\_login.html', {"msg": "Successfully Register"})

else:

return render(request, 'customer\_registration.html', {"msg": "Invalid Data"})

else:

form = CustomerForm()

return render(request, 'customer\_registration.html', {"form":form})

def customer\_logout(request):

request.session.flush()

return redirect('/customer\_login')

def admin\_home(request):

return render(request, "admin\_home.html", {})

def admin\_login(request):

if request.method == 'POST':

email = request.POST["email"]

password = request.POST["password"]

login = Admin.objects.filter(email=email, password=password)

if login.exists():

request.session['email'] = email

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

else:

return render(request, "admin\_login.html", {"msg": "Invalid Creditinals"})

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

def admin\_view\_contact(request):

contact = Contact.objects.all()

return render(request, "admin\_view\_contact.html", {"contact":contact})

def admin\_view\_customer(request):

customer = Customer.objects.all()

return render(request, "admin\_view\_customer.html", {"customer":customer})

def admin\_view\_doctors(request):

doctors = Doctors.objects.all()

return render(request, "admin\_view\_doctors.html", {"doctors":doctors})

def admin\_logout(request):

request.session.flush()

return redirect('/admin\_login')

def accept\_customer(request, id):

customer = Customer.objects.get(id=id)

customer.status = 'Accepted'

customer.save()

return redirect('/admin\_view\_customer')

def reject\_customer(request, id):

customer = Customer.objects.get(id=id)

customer.status = 'Rejected'

customer.save()

return redirect('/admin\_view\_customer')

def accept\_doctors(request, id):

doctors = Doctors.objects.get(id=id)

doctors.status = 'Accepted'

doctors.save()

return redirect('/admin\_view\_doctors')

def reject\_doctors(request, id):

doctors = Doctors.objects.get(id=id)

doctors.status = 'Rejected'

doctors.save()

return redirect('/admin\_view\_doctors')

def my\_profile(request):

email = request.session["email"]

users = Customer.objects.get(email=email)

return render(request, "my\_profile.html",{"users":users})

def my\_profile\_edit(request):

email = request.session["email"]

users = Customer.objects.get(email=email)

return render(request,"update\_my\_profile.html",{"users":users})

def update\_my\_profile(request):

if request.method == "POST":

id = request.POST['id']

users = Customer.objects.get(id=id)

form = CustomerForm(request.POST,request.FILES, instance=users)

if form.is\_valid():

form.save()

return redirect('/my\_profile') # Redirect to a view where you display updated contact details

else:

print(form.errors) # Print errors for debugging

# If not a POST request or form is not valid, render the update form again

return render(request, "update\_my\_profile.html", {})

def doctors\_home(request):

return render(request, "doctors\_home.html", {})

def doctors\_register(request):

if request.method == 'POST':

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

email = request.POST['email']

if form.is\_valid():

email=form.cleaned\_data["email"]

if Doctors.objects.filter(email=email).exists():

return render(request, 'doctors\_registration.html', {"msg": "This Email is Already Exists"})

else:

form.is\_valid()

form.save()

return render(request, 'doctors\_login.html', {"msg": "Successfully Register"})

else:

return render(request, 'doctors\_registration.html', {"msg": "Invalid Data"})

else:

form = DoctorsForm()

return render(request, 'doctors\_registration.html', {"form":form})

def doctors\_login(request):

if request.method == 'POST':

email = request.POST["email"]

password = request.POST["password"]

if Doctors.objects.filter(email=email).exists():

user = Doctors.objects.get(email=email)

if user is not None:

pwd = user.password

if password == pwd:

if user.status == "Accepted":

request.session["email"] = email

return render(request, 'doctors\_home.html', {})

else:

return render(request, "doctors\_login.html", {"msg": "Account Is On Hold"})

else:

return render(request, "doctors\_login.html", {"msg": "Wrong Passwords"})

else:

return render(request, "doctors\_login.html", {"msg": ""})

return render(request, "doctors\_login.html", {"msg": "Email Not Registered"})

return render(request, "doctors\_login.html", {})

def doctors\_profile(request):

email = request.session['email']

doctors = Doctors.objects.get(email=email)

return render(request, "doctors\_profile.html", {"doctors":doctors})

def doctors\_profile\_edit(request):

email = request.session["email"]

doctors = Doctors.objects.get(email=email)

return render(request,"doctors\_profile\_edit.html",{"doctors":doctors})

def update\_doctors\_profile(request):

if request.method == "POST":

id = request.POST['id']

doctors = Doctors.objects.get(id=id)

form = DoctorsForm(request.POST,request.FILES, instance=doctors)

if form.is\_valid():

form.save()

return redirect('/doctors\_profile') # Redirect to a view where you display updated contact details

else:

print(form.errors) # Print errors for debugging

# If not a POST request or form is not valid, render the update form again

return render(request, "doctors\_profile\_edit.html", {})

def view\_doctors(request):

doctors = Doctors.objects.filter(status='Accepted')

return render(request, "view\_doctors.html", {"doctors": doctors})

def add\_appoitments(request,id):

email = request.session['email']

doctors = Doctors.objects.get(id=id)

if request.method == "POST":

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

print(form.errors)

if form.is\_valid():

form.save()

return render(request, "add\_appoitments.html", {"msg": "Successfully Saved"})

return render(request, "add\_appoitments.html", {})

return render(request, "add\_appoitments.html", {"email":email,"doctors":doctors})

def my\_appoitments(request):

email = request.session['email']

appoitments = Appoitments.objects.filter(customer\_email=email)

return render(request,'my\_appoitments.html',{"appoitments":appoitments})

def doctors\_view\_appoitments(request):

email = request.session['email']

appoitments = Appoitments.objects.filter(doctor\_email=email)

return render(request,'doctors\_view\_appoitments.html',{"appoitments":appoitments})

def accept\_appoitments(request, id):

appoitments = Appoitments.objects.get(id=id)

appoitments.status = 'Accepted'

appoitments.save()

return redirect('/doctors\_view\_appoitments')

def reject\_appoitments(request, id):

appoitments = Appoitments.objects.get(id=id)

appoitments.status = 'Rejected'

appoitments.save()

return redirect('/doctors\_view\_appoitments')

def add\_prescriptions(request,id):

email = request.session['email']

appoitments = Appoitments.objects.get(id=id)

if request.method == "POST":

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

print(form.errors)

if form.is\_valid():

form.save()

return render(request, "add\_prescriptions.html", {"msg": "Successfully Saved"})

return render(request, "add\_prescriptions.html", {})

return render(request, "add\_prescriptions.html", {"email":email,"appoitments":appoitments})

def view\_prescriptions(request,id):

appoitments = Appoitments.objects.get(id=id)

prescriptions = Prescriptions.objects.filter(appoitments\_id=appoitments.id)

return render(request,'view\_prescriptions.html',{"prescriptions":prescriptions})

def doctor\_logout(request):

request.session.flush()

return redirect('/doctors\_login')

def deactivate(request, id):

customer = Customer.objects.get(id=id)

customer.Status = 'pending'

customer.save()

return redirect('/customer\_profile')

def is\_login(request):

if request.session.\_\_contains\_\_('email'):

return True

else:

return False

def change\_password(request):

email = request.session["email"]

print("hi")

if is\_login(request):

if request.method == 'POST':

password = request.POST["password"]

new\_password = request.POST["new\_password"]

email = request.session["email"]

try:

user = Customer.objects.get(email=email, password=password)

user.password = new\_password

user.save()

return render(request, "customer\_login.html", {"msg": "Successfully Password Updated"})

except Exception as e:

print(e, "error")

return render(request, "change\_password.html", {"email": email, "msg": "Invalid Credentials"})

return render(request, "change\_password.html", {"email": email})

def admin\_change\_password(request):

email = request.session["email"]

print("hi")

if is\_login(request):

if request.method == 'POST':

password = request.POST["password"]

new\_password = request.POST["new\_password"]

email = request.session["email"]

try:

user = Admin.objects.get(email=email, password=password)

user.password = new\_password

user.save()

return render(request, "admin\_login.html", {"msg": "Successfully Password Updated"})

except Exception as e:

print(e, "error")

return render(request, "admin\_change\_password.html", {"email": email, "msg": "Invalid Credentials"})

return render(request, "admin\_change\_password.html", {"email": email})

def add\_notifications(request):

if request.method == "POST":

form =Add\_notificationsForm(request.POST, request.FILES)

print(form.errors)

if form.is\_valid():

form.save()

return render(request, "add\_notifications.html", {"msg":"Successfully Added"})

return render(request, "add\_notifications.html", {})

return render(request, "add\_notifications.html", {})

def view\_notification(request):

notifications = Notifications.objects.all()

return render(request, "view\_notification.html", {"notifications":notifications})

def customers\_view\_notification(request):

notifications = Notifications.objects.all()

return render(request, "customers\_view\_notification.html", {"notifications":notifications})

def add\_pet(request):

email = request.session['email']

if request.method == 'POST':

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

print(form.errors)

if form.is\_valid():

form.save()

return render(request, "add\_pet.html", {"msg":"Successfully Added"})

return render(request, "add\_pet.html", {})

return render(request, "add\_pet.html", {"email":email})

def my\_pets(request):

email = request.session['email']

pets = Pet.objects.filter(email=email)

return render(request, "my\_pets.html", {"pets":pets})

def pet\_edit(request,id):

pets = Pet.objects.get(id=id)

return render(request, "pet\_edit.html", {'pets':pets})

def pet\_update(request):

if request.method == 'POST':

email = request.POST['email']

dog = Pet.objects.get(email=email)

form = PetForm(request.POST, request.FILES, instance=dog)

print(form.errors)

if form.is\_valid():

form.save()

return redirect('/my\_pets')

return render(request, "pet\_edit.html", {})

return render(request, "pet\_edit.html", {})

def delete\_view\_pet(request, id):

pets = Pet.objects.get(id=id)

pets.delete()

return redirect('/my\_pets')

def view\_all\_pets(request):

email = request.session['email']

customer = Customer.objects.get(email=email)

# Exclude the user's own pets and annotate the like count

pets = Pet.objects.exclude(email=email).annotate(like\_count=Count('pets\_likes'))

# Get the list of pet IDs that the customer has already shown interest in

interested\_pet\_ids = Interest.objects.filter(customer\_id=customer.id).values\_list('pet\_id', flat=True)

# Get the list of pet IDs that the customer has already liked

liked\_pets = Pets\_Likes.objects.filter(customer\_email=email).values\_list('pets\_id', flat=True)

return render(request, "view\_all\_pets.html", {

"pets": pets,

"interested\_pet\_ids": interested\_pet\_ids,

"liked\_pets": liked\_pets,

})

def doctor(request, id):

return render(request, "doctor.html", {})

def delete\_view\_notification(request, id):

notifications = Notifications.objects.get(id=id)

notifications.delete()

return redirect('/view\_notification')

def delete\_view\_doctor(request, id):

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

doctor.delete()

return redirect('/view\_doctor')

# views.py

def book\_now(request, id):

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

email = request.session['email']

cus = Customer.objects.get(Email=email)

if request.method == 'POST':

form = BookingForm(request.POST)

print(form.errors)

if form.is\_valid():

form.save()

return redirect('view\_bookings') # Redirect to a success page or another view

else:

form = BookingForm()

pet = Pet.objects.all()

return render(request, 'booking\_form.html', {'form': form, 'pet': pet, "x": doc, "y": cus})

# views.py

def view\_bookings(request):

book = Booking.objects.all()

return render(request, "view\_bookings.html", {"book": book})

def view\_prescription(request):

prescription = Prescription.objects.all()

return render(request, 'view\_prescription.html', {'prescription': prescription})

def view\_precautions(request):

email = request.session['email']

precautions = Precautions.objects.filter(email=email)

return render(request, 'view\_precautions.html', {'precautions':precautions})

def customers\_view\_precautions(request):

precautions = Precautions.objects.all()

return render(request, 'customers\_view\_precautions.html', {'precautions':precautions})

def add\_precautions(request):

email = request.session['email']

if request.method == "POST":

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

print(form.errors)

if form.is\_valid():

form.save()

return render(request, "add\_precautions.html", {"msg": "Successfully Saved"})

return render(request, "add\_precautions.html", {})

return render(request, "add\_precautions.html", {"email":email})

def quote\_view(request):

doctor = Quote.objects.all()

return render(request, 'view\_quote.html', {'doctor': doctor})

def add\_quote(request):

if request.method == 'POST':

form = QuoteForm(request.POST)

if form.is\_valid():

form.save()

return redirect('quote\_view')

else:

form = QuoteForm()

return render(request, 'add\_quote.html', {'form': form})

def add\_reviews(request,id):

email = request.session['email']

doctors = Doctors.objects.get(id=id)

if request.method == 'POST':

form = ReviewsForm(request.POST)

if form.is\_valid():

form.save()

return render(request, 'add\_reviews.html', {"msg":"Successfully Saved"})

else:

form = ReviewsForm()

return render(request, 'add\_reviews.html', {'form': form,"doctors":doctors,"email":email})

def view\_reviews(request,id):

doctors = Doctors.objects.get(id=id)

reviews = Reviews.objects.filter(doctors\_id=doctors.id)

return render(request, 'view\_reviews.html', {'reviews':reviews})

def doctors\_view\_reviews(request):

email = request.session['email']

doctors = Doctors.objects.get(email=email)

reviews = Reviews.objects.filter(doctors\_id=doctors.id)

return render(request, 'doctors\_view\_reviews.html', {'reviews':reviews})

def doctors\_change\_password(request):

email = request.session["email"]

print("hi")

if is\_login(request):

if request.method == 'POST':

password = request.POST["password"]

new\_password = request.POST["new\_password"]

email = request.session["email"]

try:

user = Doctors.objects.get(email=email, password=password)

user.password = new\_password

user.save()

return render(request, "doctors\_login.html", {"msg": "Successfully Password Updated"})

except Exception as e:

print(e, "error")

return render(request, "doctors\_change\_password.html", {"email": email, "msg": "Invalid Credentials"})

return render(request, "doctors\_change\_password.html", {"email": email})

def view\_all\_customers(request):

email = request.session['email']

customers = Customer.objects.exclude(email=email)

# Fetch the friend requests where the logged-in user is either the sender or receiver

sent\_requests = Friends\_Requests.objects.filter(from\_email=email)

received\_requests = Friends\_Requests.objects.filter(to\_email=email)

# Combine both sender and receiver requests

requested\_ids = list(sent\_requests.values\_list('to\_email', flat=True)) + list(received\_requests.values\_list('from\_email', flat=True))

# Fetch accepted friend requests (either direction)

accepted\_requests = Friends\_Requests.objects.filter(

(Q(from\_email=email) | Q(to\_email=email)),

status='Accepted'

).values\_list('from\_email', 'to\_email')

# Convert accepted requests into a list of emails for easier lookup

friends\_ids = [request[0] if request[1] == email else request[1] for request in accepted\_requests]

return render(request, 'view\_all\_customers.html', {

"customers": customers,

"requested\_ids": requested\_ids,

"friends\_ids": friends\_ids

})

def add\_friends\_requests(request, id):

email = request.session['email']

customer = Customer.objects.get(id=id)

# Check if the request already exists

if Friends\_Requests.objects.filter(from\_email=email, to\_email=customer.email).exists():

return JsonResponse({"msg": "Request Already Sent"}, status=400)

if request.method == "POST":

# Create the friend request

Friends\_Requests.objects.create(

customers=customer,

from\_email=email,

to\_email=customer.email

)

return JsonResponse({"msg": "Friend Request Sent"}, status=200)

return JsonResponse({"msg": "Invalid Request"}, status=400)

def view\_friends\_requests(request):

email = request.session['email']

requests = Friends\_Requests.objects.filter(to\_email=email)

return render(request,'view\_friends\_requests.html',{"requests":requests})

def accept\_friends\_requests(request, id):

requests = Friends\_Requests.objects.get(id=id)

requests.status = 'Accepted'

requests.save()

return redirect('/view\_friends\_requests')

def reject\_friends\_requests(request, id):

requests = Friends\_Requests.objects.get(id=id)

requests.status = 'Rejected'

requests.save()

return redirect('/view\_friends\_requests')

def my\_friends(request):

email = request.session['email']

# Fetching friends where the logged-in user is either the sender or receiver, and the request is accepted

sent\_friends = Friends\_Requests.objects.filter(from\_email=email, status='Accepted')

received\_friends = Friends\_Requests.objects.filter(to\_email=email, status='Accepted')

return render(request, 'my\_friends.html', {

"sent\_friends": sent\_friends,

"received\_friends": received\_friends

})

def add\_posts(request):

email = request.session['email']

if request.method == "POST":

form = Add\_PostsForms(request.POST, request.FILES)

print(form.errors)

if form.is\_valid():

form.save()

return render(request, "add\_posts.html", {"msg": "Successfully Saved"})

return render(request, "add\_posts.html", {})

return render(request, "add\_posts.html", {"email":email})

def admin\_view\_posts(request):

posts = Add\_Posts.objects.all()

return render(request,'admin\_view\_posts.html',{"posts":posts})

def accept\_posts(request, id):

posts = Add\_Posts.objects.get(id=id)

posts.status = 'Accepted'

posts.save()

return redirect('/admin\_view\_posts')

def reject\_posts(request, id):

posts = Add\_Posts.objects.get(id=id)

posts.status = 'Rejected'

posts.save()

return redirect('/admin\_view\_posts')

def my\_posts(request):

email = request.session['email']

posts = Add\_Posts.objects.filter(email=email)

return render(request,'my\_posts.html',{"posts":posts})

def view\_posts(request):

email = request.session['email']

# Get the user's friends (accepted friend requests)

friend\_pairs = Friends\_Requests.objects.filter(

(Q(from\_email=email) | Q(to\_email=email)),

status='Accepted'

).values\_list('from\_email', 'to\_email')

# Flatten the friend pairs into a single list of emails, excluding the user's own email

friends = set(chain(\*friend\_pairs)) - {email}

# Get posts visible to all or only to friends if they are in the friend list

posts = Add\_Posts.objects.filter(

Q(only\_for='All') |

(Q(only\_for='Only\_Friends') & Q(email\_\_in=friends)),

status='Accepted'

).exclude(email=email).annotate(like\_count=Count('likes')) # Count likes for each post

# Get the posts liked by the user

liked\_posts = Likes.objects.filter(customer\_email=email).values\_list('post\_id', flat=True)

return render(request, 'view\_posts.html', {

"posts": posts,

"liked\_posts": liked\_posts,

})

def posts\_edit(request,id):

posts = Add\_Posts.objects.get(id=id)

return render(request, "posts\_edit.html", {'posts':posts})

def posts\_update(request):

if request.method == 'POST':

id = request.POST['id']

posts = Add\_Posts.objects.get(id=id)

form = Add\_PostsForms(request.POST, request.FILES, instance=posts)

print(form.errors)

if form.is\_valid():

form.save()

return redirect('/my\_posts')

return render(request, "posts\_edit.html", {})

return render(request, "posts\_edit.html", {})

def delete\_posts(request, id):

posts = Add\_Posts.objects.get(id=id)

posts.delete()

return redirect('/my\_posts')

def like\_post(request, post\_id):

email = request.session['email']

post = get\_object\_or\_404(Add\_Posts, id=post\_id)

# Check if the user has already liked the post

if not Likes.objects.filter(post=post, customer\_email=email).exists():

# Add the like

Likes.objects.create(post=post, customer\_email=email, date\_time=timezone.now())

msg = "Liked successfully."

else:

msg = "You've already liked this post."

return redirect('view\_posts')

def view\_posts\_likes(request,id):

posts = Add\_Posts.objects.get(id=id)

likes = Likes.objects.filter(post\_id=posts.id)

return render(request,'view\_posts\_likes.html',{"likes":likes})

def add\_comments(request,id):

email = request.session['email']

posts = Add\_Posts.objects.get(id=id)

comments = Add\_Comments.objects.filter(posts\_id=posts.id)

if request.method == "POST":

form = Add\_CommentsForm(request.POST, request.FILES)

print(form.errors)

if form.is\_valid():

form.save()

return render(request, "add\_comments.html", {"msg": "Successfully Saved","comments":comments})

return render(request, "add\_comments.html", {"comments":comments})

return render(request, "add\_comments.html", {"email":email,"posts":posts,"comments":comments})

def view\_posts\_comments(request,id):

posts = Add\_Posts.objects.get(id=id)

comments = Add\_Comments.objects.filter(posts\_id=posts.id)

return render(request,'view\_posts\_comments.html',{"comments":comments})

def express\_interest(request, pet\_id):

email = request.session['email']

# Get the customer object based on the session email

customer = Customer.objects.get(email=email)

# Check if the user has already expressed interest in this pet

if not Interest.objects.filter(customer=customer, pet\_id=pet\_id).exists():

# Get the pet and save the interest

pet = Pet.objects.get(id=pet\_id)

Interest.objects.create(

customer=customer,

pet=pet,

interest\_date=timezone.now()

)

return redirect('view\_all\_pets')

def toggle\_sell\_status(request, id):

# Get the pet by ID

pet = Pet.objects.get(id=id)

# Toggle the sell\_for field

if pet.sell\_for == 'Yes':

pet.sell\_for = 'No'

else:

pet.sell\_for = 'Yes'

pet.save()

# Redirect to 'my\_pets' after updating the status

return redirect('my\_pets')

def view\_pets\_requests(request,id):

pets = Pet.objects.get(id=id)

requests = Interest.objects.filter(pet\_id=pets.id)

return render(request,'view\_pets\_requests.html',{"requests":requests})

def like\_pets(request, pets\_id):

email = request.session['email']

pets = get\_object\_or\_404(Pet, id=pets\_id)

# Check if the user has already liked the post

if not Pets\_Likes.objects.filter(pets=pets, customer\_email=email).exists():

# Add the like

Pets\_Likes.objects.create(pets=pets, customer\_email=email, date\_time=timezone.now())

msg = "Liked successfully."

else:

msg = "You've already liked this post."

return redirect('view\_all\_pets')

def view\_pets\_likes(request,id):

pets = Pet.objects.get(id=id)

likes = Pets\_Likes.objects.filter(pets\_id=pets.id)

return render(request,'view\_pets\_likes.html',{"likes":likes})

def doctors\_view\_notifications(request):

notifications = Notifications.objects.all()

return render(request, "doctors\_view\_notifications.html", {"notifications": notifications})

**urls.py:**

"""

URL configuration for pets\_cares\_owners\_communication\_systems project.

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

https://docs.djangoproject.com/en/5.0/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 import settings

from django.conf.urls.static import static

from django.contrib import admin

from django.urls import path

from pets\_cares\_owners\_communication\_systems\_app import views

urlpatterns = [

path('admin/', admin.site.urls),

path('', views.index,name="index"),

path('about',views.about,name="about"),

path('blog',views.blog,name="blog"),

path('blog\_details',views.blog\_details,name="blog\_details"),

path('contact',views.contact,name="contact"),

path('services',views.services,name="services"),

path('customer\_login',views.customer\_login,name="customer\_login"),

path('customer\_register',views.customer\_register,name="customer\_register"),

path('customer\_home',views.customer\_home,name="customer\_home"),

path('my\_profile',views.my\_profile,name="my\_profile"),

path('customer\_logout',views.customer\_logout,name="customer\_logout"),

path('admin\_home',views.admin\_home,name="admin\_home"),

path('admin\_view\_contact',views.admin\_view\_contact,name="admin\_view\_contact"),

path('admin\_view\_customer',views.admin\_view\_customer,name="admin\_view\_customer"),

path('admin\_logout',views.admin\_logout,name="admin\_logout"),

path('admin\_login', views.admin\_login, name="admin\_login"),

path('accept\_customer/<int:id>',views.accept\_customer,name="accept\_customer"),

path('reject\_customer/<int:id>',views.reject\_customer,name="reject\_customer"),

path('my\_profile\_edit',views.my\_profile\_edit,name="my\_profile\_edit"),

path('update\_my\_profile',views.update\_my\_profile,name="update\_my\_profile"),

path('doctors\_home',views.doctors\_home,name="doctors\_home"),

path('doctors\_profile',views.doctors\_profile,name="doctors\_profile"),

path('doctors\_register',views.doctors\_register,name="doctors\_register"),

path('doctors\_login',views.doctors\_login,name="doctors\_login"),

path('doctor\_logout',views.doctor\_logout,name="doctor\_logout"),

path('accept\_appoitments/<int:id>',views.accept\_appoitments,name="accept\_appoitments"),

path('reject\_appoitments/<int:id>',views.reject\_appoitments,name="reject\_appoitments"),

path('change\_password',views.change\_password,name="change\_password"),

path('admin\_change\_password',views.admin\_change\_password,name="admin\_change\_password"),

path('deactivate/<int:id>',views.deactivate,name="deactivate"),

path('add\_notifications',views.add\_notifications,name="add\_notifications"),

path('view\_notification', views.view\_notification, name="view\_notification"),

path('add\_pet',views.add\_pet,name="add\_pet"),

# path('templates',views.my\_pets,name="templates"),

path('add\_pet',views.add\_pet,name="add\_pet"),

path('my\_pets',views.my\_pets,name="my\_pets"),

path('view\_doctors', views.view\_doctors, name="view\_doctors"),

path('delete\_view\_notification/<int:id>',views.delete\_view\_notification,name="delete\_view\_notification"),

path('delete\_view\_pet/<int:id>', views.delete\_view\_pet, name="delete\_view\_pet"),

path('pet\_edit/<int:id>',views.pet\_edit,name="pet\_edit"),

path('delete\_view\_doctor/<int:id>', views.delete\_view\_doctor, name="delete\_view\_doctor"),

path('pet\_update',views.pet\_update,name="pet\_update"),

path('book\_now/<int:id>',views.book\_now,name="book\_now"),

path('view\_bookings', views.view\_bookings, name="view\_bookings"),

path('add\_prescriptions/<int:id>', views.add\_prescriptions, name="add\_prescriptions"),

path('view\_precautions', views.view\_precautions, name="view\_precautions"),

path('add\_precautions', views.add\_precautions, name="add\_precautions"),

path('quote\_view', views.quote\_view, name='quote\_view'),

path('add\_quote',views.add\_quote, name="add\_quote"),

path('add\_reviews/<int:id>', views.add\_reviews, name="add\_reviews"),

path('view\_reviews/<int:id>', views.view\_reviews, name="view\_reviews"),

path('doctors\_view\_reviews', views.doctors\_view\_reviews, name='doctors\_view\_reviews'),

path('view\_prescription',views.view\_prescription, name='view\_prescription'),

path('customers\_view\_notification', views.customers\_view\_notification, name="customers\_view\_notification"),

path('view\_all\_pets', views.view\_all\_pets, name='view\_all\_pets'),

path('admin\_view\_doctors', views.admin\_view\_doctors, name='admin\_view\_doctors'),

path('accept\_doctors/<int:id>', views.accept\_doctors, name="accept\_doctors"),

path('reject\_doctors/<int:id>', views.reject\_doctors, name="reject\_doctors"),

path('doctors\_profile\_edit', views.doctors\_profile\_edit, name='doctors\_profile\_edit'),

path('update\_doctors\_profile', views.update\_doctors\_profile, name='update\_doctors\_profile'),

path('doctors\_change\_password', views.doctors\_change\_password, name='doctors\_change\_password'),

path('customers\_view\_precautions', views.customers\_view\_precautions, name='customers\_view\_precautions'),

path('add\_appoitments/<int:id>', views.add\_appoitments, name='add\_appoitments'),

path('my\_appoitments', views.my\_appoitments, name='my\_appoitments'),

path('doctors\_view\_appoitments', views.doctors\_view\_appoitments, name='doctors\_view\_appoitments'),

path('view\_prescriptions/<int:id>', views.view\_prescriptions, name='view\_prescriptions'),

path('view\_all\_customers', views.view\_all\_customers, name='view\_all\_customers'),

path('add\_friends\_requests/<int:id>', views.add\_friends\_requests, name='add\_friends\_requests'),

path('view\_friends\_requests', views.view\_friends\_requests, name='view\_friends\_requests'),

path('accept\_friends\_requests/<int:id>', views.accept\_friends\_requests, name='accept\_friends\_requests'),

path('reject\_friends\_requests/<int:id>', views.reject\_friends\_requests, name='reject\_friends\_requests'),

path('my\_friends', views.my\_friends, name='my\_friends'),

path('add\_posts', views.add\_posts, name='add\_posts'),

path('my\_posts', views.my\_posts, name='my\_posts'),

path('view\_posts', views.view\_posts, name='view\_posts'),

path('admin\_view\_posts', views.admin\_view\_posts, name='admin\_view\_posts'),

path('accept\_posts/<int:id>', views.accept\_posts, name='accept\_posts'),

path('reject\_posts/<int:id>', views.reject\_posts, name='reject\_posts'),

path('posts\_edit/<int:id>', views.posts\_edit, name='posts\_edit'),

path('posts\_update', views.posts\_update, name='posts\_update'),

path('delete\_posts/<int:id>', views.delete\_posts, name='delete\_posts'),

path('like\_post/<int:post\_id>', views.like\_post, name='like\_post'),

path('view\_posts\_likes/<int:id>', views.view\_posts\_likes, name='view\_posts\_likes'),

path('add\_comments/<int:id>', views.add\_comments, name='add\_comments'),

path('view\_posts\_comments/<int:id>', views.view\_posts\_comments, name='view\_posts\_comments'),

path('express\_interest/<int:pet\_id>', views.express\_interest, name='express\_interest'),

path('toggle\_sell\_status/<int:id>', views.toggle\_sell\_status, name='toggle\_sell\_status'),

path('view\_pets\_requests/<int:id>', views.view\_pets\_requests, name='view\_pets\_requests'),

path('like\_pets/<int:pets\_id>', views.like\_pets, name='like\_pets'),

path('view\_pets\_likes/<int:id>', views.view\_pets\_likes, name='view\_pets\_likes'),

path('doctors\_view\_notifications', views.doctors\_view\_notifications, name='doctors\_view\_notifications'),

]

if settings.DEBUG:

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

**templates:**

{% load static %}

{% include "header.html" %}

<!--================ Hero sm Banner start =================-->

<section class="hero-banner hero-banner--sm">

<div class="hero-banner\_\_content text-center">

<h1>Admin Login</h1>

<nav aria-label="breadcrumb" class="banner-breadcrumb">

</nav>

</div>

</section>

<!--================ Hero sm Banner end =================-->

<!-- ================ contact section start ================= -->

<section class="section-margin--large">

<div class="container">

<div class="col-md-8 col-lg-9">

<div style="color:red">{{msg}}</div>

<form action="" class="form-contact contact\_form" method="post" id="contactForm">

{% csrf\_token %}

<div class="row">

<div class="col-lg-9">

<!-- <p>Please Login</p>-->

<div class="form-group">

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

</div>

<div class="form-group">

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

</div>

<div class="form-group text-center text-md-left mt-3">

<button type="submit" class="button button-contactForm">Login</button>

</div>

</div>

</div>

</form>

</div>

</div>

</section>

<!-- ================ contact section end ================= -->

{% include "footer.html" %}

{% load static %}

{% include "header.html" %}

<!--================ Hero sm Banner start =================-->

<section class="hero-banner hero-banner--sm">

<div class="hero-banner\_\_content text-center">

<h1>Customer Login </h1>

<nav aria-label="breadcrumb" class="banner-breadcrumb">

</nav>

</div>

</section>

<!--================ Hero sm Banner end =================-->

<!-- ================ contact section start ================= -->

<section class="section-margin--large">

<div class="container">

<div class="d-none d-sm-block mb-5 pb-4">

<div style="color:red">{{msg}}</div>

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

{% csrf\_token %}

<div class="row">

<div class="col-lg-5">

<div class="form-group">

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

</div>

<div class="form-group">

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

</div>

<div class="form-group">

<button type="submit" value="Login" class="button button-contactForm">Login</button>

If You Are A New Customers <a href="{% url 'customer\_register' %}" style="color:red">Register</a>

</div>

</div>

</div>

</form>

</div>

</div>

</section>

<!-- ================ contact section end ================= -->

{% include "footer.html" %}

{% load static %}

{% include "header.html" %}

<!--================ Hero sm Banner start =================-->

<section class="hero-banner hero-banner--sm">

<div class="hero-banner\_\_content text-center">

<h1>Doctors Login </h1>

<nav aria-label="breadcrumb" class="banner-breadcrumb">

</nav>

</div>

</section>

<!--================ Hero sm Banner end =================-->

<!-- ================ contact section start ================= -->

<section class="section-margin--large">

<div class="container">

<div class="d-none d-sm-block mb-5 pb-4">

<div style="color:red">{{msg}}</div>

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

{% csrf\_token %}

<div class="row">

<div class="col-lg-5">

<div class="form-group">

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

</div>

<div class="form-group">

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

</div>

<div class="form-group">

<button type="submit" value="Login" class="button button-contactForm">Login</button>

If You Are A New Customers <a href="{% url 'doctors\_register' %}" style="color:red">Register</a>

</div>

</div>

</div>

</form>

</div>

</div>

</section>

<!-- ================ contact section end ================= -->

{% include "footer.html" %}