# Complaint management

Models.py:

from django.db import models

# 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.CharField(max\_length=100)

class Meta:

db\_table = "contact"

class Customer(models.Model):

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

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

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

mobile = models.BigIntegerField()

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

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

class Meta:

db\_table = "customer"

class Administration(models.Model):

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

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

class Meta:

db\_table = "administration"

class AddCatagory(models.Model):

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

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

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

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

status = models.IntegerField(default=0)

class Meta:

db\_table = "addcatagory"

class Solvers(models.Model):

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

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

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

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

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

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

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

class Meta:

db\_table = "Solvers"

class Update\_Status(models.Model):

addcatagory = models.ForeignKey(AddCatagory, on\_delete=models.CASCADE)

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

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

class Meta:

db\_table = "Update\_Status"

class Send\_Query(models.Model):

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

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

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

class Meta:

db\_table = "Send\_Query"

class Replay(models.Model):

solvers = models.ForeignKey(Solvers, on\_delete=models.CASCADE)

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

class Meta:

db\_table = "Replay"

views.py:

from django.shortcuts import render, redirect

from complaint\_managementapp.models import Contact, Customer, Administration, AddCatagory, Solvers, Update\_Status, \

Send\_Query, Replay

from complaint\_managementapp.forms import ContactForm, CustomerForm, AdministrationForm, AddCatagoryForm, SolversForm, \

Update\_StatusForm, Send\_QueryForm, ReplayForm

# 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\_single(request):

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

def main(request):

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

def rooms(request):

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

def services(request):

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

def contact(request):

if request.method == "POST":

print("hii")

form = ContactForm(request.POST)

print("hii")

if form.is\_valid():

print("hello")

print("hloo")

form.save()

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

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

# Customer Views

def customer(request):

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

def regpage(request):

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

def customer\_reg(request):

if request.method == "POST":

form = CustomerForm(request.POST)

print(form.errors)

if form.is\_valid():

try:

form.save()

except Exception as e:

print(e)

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

def is\_customer\_login(request):

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

return True

else:

return False

def customer\_login(request):

if request.method == "POST":

email = request.POST['email']

password = request.POST['password']

print(email, " ", password)

customer = Customer.objects.filter(email=email, password=password)

if customer.exists():

request.session["email"] = email

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

else:

return render(request, "customer\_login.html", {"msg": "email or password not exist"})

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

def customer\_home(request):

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

def customer\_change(request):

if is\_customer\_login(request):

if request.method == "POST":

email = request.session["email"]

password = request.POST["password"]

newpassword = request.POST["newpassword"]

try:

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

customer.password = newpassword

customer.save()

msg = "password updated successfully"

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

except:

msg = "inavlid data"

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

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

else:

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

def customer\_display(request):

customer = Customer.objects.all()

print("hello")

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

def customer\_delete(requedef, id):

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

customer.delete()

return redirect("/customer\_reg")

def customer\_edit(request, id):

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

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

# if customer.is\_valid():

# customer.save()

# return redirect("/customer\_display")

def addcatagory(request):

if request.method == "POST":

addcatagory = AddCatagoryForm(request.POST, request.FILES)

if addcatagory.is\_valid():

addcatagory.save()

return render(request, "addcatagory.html", {"msg": "success"})

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

def addcatagory\_delete(request, id):

addcatagory = AddCatagory.objects.get(id=id)

addcatagory.delete()

return redirect("/addcatagory")

def addcatagory\_edit(request, id):

addcatagory = AddCatagory.objects.get(id=id)

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

def customer\_update(request):

if request.method == "POST":

customerid = request.POST["id"]

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

customer = CustomerForm(request.POST, instance=customer)

if customer.is\_valid():

customer.save()

return redirect("/customer\_display")

return redirect("/customer\_display")

def addcatagory\_update(request):

if request.method == "POST":

addcatagoryid = request.POST["id"]

addcatagory = AddCatagory.objects.get(id=addcatagoryid)

addcatagory = AddCatagoryForm(request.POST, instance=addcatagory)

if addcatagory.is\_valid():

addcatagory.save()

return redirect("/customer\_viewcatagory")

return redirect("/customer\_viewcatagory")

def accept\_addcatagory(request, id):

accept = AddCatagory.objects.get(id=id)

accept.status = 1

accept.save()

return redirect("/admin\_viewcatagory")

def reject\_addcatagory(request, id):

reject = AddCatagory.objects.get(id=id)

reject.status = 2

reject.save()

return redirect("/admin\_viewcatagory")

def cancel\_addcatagory(request, id):

reject = AddCatagory.objects.get(id=id)

reject.status = 3

reject.save()

return render(request, "customerview\_catagory.html", {})

def customer\_viewcatagory(request):

addcatagory = AddCatagory.objects.all()

print("hi")

return render(request, "customer\_viewcatagory.html", {"addcatagory": addcatagory})

def customer\_send\_query(request):

if request.method == "POST":

send\_query = Send\_QueryForm(request.POST)

if send\_query.is\_valid():

send\_query.save()

return render(request, "customer\_send\_query.html", {"msg": "success"})

solvers = Solvers.objects.all()

return render(request, "customer\_send\_query.html", {"solvers": solvers})

def view\_update\_status(request):

update\_status = Update\_Status.objects.all()

print("hello")

return render(request, "view\_update\_status.html", {"update\_status": update\_status})

def customer\_view\_replay(request, id):

solvers = Solvers.objects.get(id=id)

replay = Replay.objects.filter(solvers\_id=solvers.id)

print("hello")

return render(request, "customer\_view\_replay.html", {"replay": replay})

def customer\_view\_query(request):

send\_query = Send\_Query.objects.all()

print("hello")

return render(request, "customer\_view\_query.html", {"send\_query": send\_query})

def solvers\_view\_query(request):

send\_query = Send\_Query.objects.all()

print("hello")

return render(request, "solvers\_view\_query.html", {"send\_query": send\_query})

def customer\_logout(request):

if request.session.has\_key("email"):

email = request.session["email"]

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

# Admin Views

def administration(request):

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

def admin\_login(request):

if request.method == "POST":

email = request.POST['email']

password = request.POST['password']

print(email, " ", password)

administration = Administration.objects.filter(email=email, password=password)

if administration.exists():

print("hello")

request.session["email"] = email

return render(request, "administration.html", {"msg": email})

else:

return render(request, "admin\_login.html", {"msg": "email or password not exist"})

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

def admin\_home(request):

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

def is\_admin\_login(request):

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

return True

else:

return False

def admin\_change(request):

if is\_admin\_login(request):

if request.method == "POST":

email = request.session["email"]

password = request.POST["password"]

newpassword = request.POST["newpassword"]

try:

administration = Administration.objects.get(email=email, password=password)

administration.password = newpassword

administration.save()

msg = "password updated successfully"

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

except:

msg = "inavlid data"

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

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

else:

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

def admin\_logout(request):

if request.session.has\_key("email"):

email = request.session["email"]

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

def update\_status(request, id):

addcatagory = AddCatagory.objects.get(id=id)

if request.method == "POST":

form = Update\_StatusForm(request.POST)

if form.is\_valid():

form.save()

return render(request, "update\_status.html", {"msg": "success", "addcatagory": addcatagory.id})

return render(request, "update\_status.html", {"addcatagory": addcatagory.id})

def admin\_viewcustomer(request):

customer = Customer.objects.all()

print("hello")

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

def admin\_viewcontact(request):

contact = Contact.objects.all()

print("hello")

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

def admin\_viewquery(request):

send\_query = Send\_Query.objects.all()

print("hello")

return render(request, "admin\_view\_query.html", {"send\_query": send\_query})

def admin\_viewcatagory(request):

addcatagory = AddCatagory.objects.all()

print("hello")

return render(request, "admin\_view\_catagory.html", {"addcatagory": addcatagory})

# manager views

def solvers(request):

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

def solvers\_regpage(request):

return render(request, "solvers\_reg.html", {})

def solvers\_reg(request):

if request.method == "POST":

form = SolversForm(request.POST)

print(form.errors)

if form.is\_valid():

try:

form.save()

except Exception as e:

print(e)

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

def solvers\_login(request):

if request.method == "POST":

email = request.POST['email']

password = request.POST['password']

print(email, " ", password)

solvers = Solvers.objects.filter(email=email, password=password)

if solvers.exists():

request.session["email"] = email

return render(request, "solvers.html", {"msg": email})

else:

return render(request, "solvers\_login.html", {"msg": "email or password not exist"})

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

def solver\_home(request):

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

def is\_solvers\_login(request):

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

return True

else:

return False

def solvers\_change(request):

if is\_solvers\_login(request):

if request.method == "POST":

email = request.session["email"]

password = request.POST["password"]

newpassword = request.POST["newpassword"]

try:

solvers = Solvers.objects.get(email=email, password=password)

solvers.password = newpassword

solvers.save()

msg = "password updated successfully"

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

except:

msg = "inavlid data"

return render(request, "solvers\_change.html", {"msg": msg})

return render(request, "solvers\_change.html", {})

else:

return render(request, "solvers\_change.html", {})

def solvers\_display(request):

solvers = Solvers.objects.all()

print("hello")

return render(request, "solvers\_display.html", {"solvers": solvers})

def solvers\_edit(request, id):

solvers = Solvers.objects.get(id=id)

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

def solvers\_update(request):

if request.method == "POST":

solversid = request.POST["id"]

solvers = Solvers.objects.get(id=solversid)

solvers = SolversForm(request.POST, instance=solvers)

if solvers.is\_valid():

solvers.save()

return redirect("/solvers\_display")

return redirect("/solvers\_display")

def solvers\_delete(request, id):

solvers = Solvers.objects.get(id=id)

solvers.delete()

return redirect("/solvers\_reg")

def solvers\_view\_query(request):

send\_query = Send\_Query.objects.all()

print("hello")

return render(request, "solvers\_view\_query.html", {"send\_query": send\_query})

def solvers\_replay(request, id):

solvers = Solvers.objects.get(id=id)

if request.method == "POST":

replay = ReplayForm(request.POST)

if replay.is\_valid():

print("errors", replay.errors)

replay.save()

return render(request, "solvers\_replay.html", {"msg": "success", "solvers": solvers.id})

return render(request, "solvers\_replay.html", {"solvers": solvers.id})

def solvers\_logout(request):

if request.session.has\_key("email"):

email = request.session["email"]

return render(request, "solvers\_login.html", {"email": email})

urls.py:

from django.contrib import admin

from django.urls import path

from complaint\_managementapp import views as v

urlpatterns = [

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

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

path('about/', v.about, name="about"),

path('blog/', v.blog, name="blog"),

path('blog\_single/', v.blog\_single, name="blog\_single"),

path('main/', v.main, name="main"),

path('rooms/', v.rooms, name="rooms"),

path('services/', v.services, name="services"),

path('contact/', v.contact, name="contact"),

# Customer urls

path('customer/', v.customer, name="customer"),

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

path('regpage/', v.regpage, name="regpage"),

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

path('customer\_reg/', v.customer\_reg, name="customer\_reg"),

path('customer\_change/', v.customer\_change, name="customer\_change"),

path('customer\_display/', v.customer\_display, name="customer\_display"),

path('customer\_update/', v.customer\_update, name="customer\_update"),

path('customer\_delete/<int:id>', v.customer\_delete, name="customer\_delete"),

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

path('addcatagory\_update/', v.addcatagory\_update, name="addcatagory\_update"),

path('addcatagory\_delete/<int:id>', v.addcatagory\_delete, name="addcatagory\_delete"),

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

path('addcatagory/', v.addcatagory, name="addcatagory"),

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

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

path('cancel\_addcatagory/<int:id>', v.cancel\_addcatagory, name="cancel\_addcatagory"),

path('view\_update\_status/', v.view\_update\_status, name="view\_update\_status"),

path('customer\_send\_query/', v.customer\_send\_query, name="customer\_send\_query"),

path('customer\_view\_replay/<int:id>', v.customer\_view\_replay, name="customer\_view\_replay"),

path('customer\_view\_query/', v.customer\_view\_query, name="customer\_view\_query"),

path('customer\_viewcatagory/', v.customer\_viewcatagory, name="customer\_viewcatagory"),

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

# admin urls

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

path('admin\_change/', v.admin\_change, name="admin\_change"),

path('regpage/', v.regpage, name="regpage"),

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

path('admin\_viewcontact/', v.admin\_viewcontact, name="admin\_viewcontact"),

path('admin\_viewcustomer/', v.admin\_viewcustomer, name="admin\_viewcustomer"),

path('admin\_viewcatagory/', v.admin\_viewcatagory, name="admin\_viewcatagory"),

path('admin\_viewquery/', v.admin\_viewquery, name="admin\_viewquery"),

path('update\_status/<int:id>', v.update\_status, name="update\_status"),

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

# solver urls

path('solvers/', v.solvers, name="solvers"),

path('solvers\_login/', v.solvers\_login, name="solvers\_login"),

path('solvers\_regpage/', v.solvers\_regpage, name="solvers\_regpage"),

path('solvers\_reg/', v.solvers\_reg, name="solvers\_reg"),

path('solver\_home/', v.solver\_home, name="solver\_home"),

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

path('solvers\_update/', v.solvers\_update, name="solvers\_update"),

path('solvers\_delete/<int:id>', v.solvers\_delete, name="solvers\_delete"),

path('solvers\_change/', v.solvers\_change, name="solvers\_change"),

path('solvers\_display/', v.solvers\_display, name="solvers\_display"),

path('solvers\_view\_query/', v.solvers\_view\_query, name="solvers\_view\_query"),

path('solvers\_replay/<int:id>', v.solvers\_replay, name="solvers\_replay"),

path('solvers\_logout/', v.solvers\_logout, name="solvers\_logout"),

]

Templates:

Customer\_register.html

{% load static %}

{% include 'header.html' %}

<section class="hero-wrap hero-wrap-2" style="background-image: url('{% static 'images/bg\_2.jpg' %}');" data-stellar-background-ratio="0.5">

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

<div class="container">

<div class="row no-gutters slider-text align-items-center justify-content-center">

<div class="col-md-9 ftco-animate text-center">

<h1 class="mb-0 bread">Customer Register</h1>

</div>

</div>

</div>

</section>

<br>

<form align="center" method ="POST" class="post-form" action="/customer\_reg/">

{% csrf\_token %}

<div class="container h-100">

<h3 class="fs-4 card-title fw-bold mb-4">Register Here</h3>

<div class="mb-2 w-100">

<label class="text-muted" for="email">Email</label>

<input id="email" type="email" name="email" required>

</div>

<div class="mb-2 w-100">

<label class="text" for="fullname">Fullname</label>

<input id="fullname" type="text" name="fullname" required>

</div>

<div class="mb-2 w-100">

<label class="text-muted" for="password">Password</label>

<input id="password" type="password" name="password" required>

</div>

<div class="mb-2 w-100">

<label class="text" for="mobile">Mobile</label>

<input id="mobile" type="number" name="mobile" required>

</div>

<div class="mb-2 w-100">

<label class="text" for="address">Address</label>

<input id="address" type="text" name="address" required>

</div>

<div class="mb-2 w-100" >

<label class="text" for="city">City</label>

<select name="city" id="city">

<option value="Hyd">Hyd</option>

<option value="Pune">Pune</option>

<option value="Mumbai">Mumbai</option>

</select>

</div>

<button type="submit" class="btn btn-primary ms-auto">Register</button><br>

</div></form>

{% include 'footer.html' %}

Customer\_login.html:

{% load static %}

{% include 'header.html' %}

<section class="hero-wrap hero-wrap-2" style="background-image: url('{% static 'images/bg\_2.jpg' %}');" data-stellar-background-ratio="0.5">

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

<div class="container">

<div class="row no-gutters slider-text align-items-center justify-content-center">

<div class="col-md-9 ftco-animate text-center">

<h1 class="mb-0 bread">Customer Login</h1>

</div>

</div>

</div>

</section>

<form align = "center" method ="POST" class="post-form" action="/customer\_login/">

{% csrf\_token %}

<div class="container h-100"><br>

<h3 class="fs-4 card-title fw-bold mb-4">Login Here</h3>

<div class="mb-2 w-100">

<label class="col-sm-2 col-form-label" for="Email">Email</label>

<input id="Email" type="email" name="email" required>

</div><br>

<div class="mb-2 w-100">

<label class="col-sm-2 col-form-label" for="password">Password</label>

<input id="password" type="password" name="password" required>

</div>

<button type="submit" class="btn btn-primary ">Login</button><br>

U are not exist? <a href="/regpage/">Register</a>

</div>

</form>

{% include 'footer.html' %}