CONNECTING SOCIAL MEDIA ECOMMERCE

VIEWS:

from itertools import chain  
  
from django.contrib import messages  
from django.db.models import Q, Count  
from django.shortcuts import render, redirect, get\_object\_or\_404  
from django.utils import timezone  
  
from .models import Admin, Customer, Vendor, Notification, Category, Product, Cart, Orders, Order\_Products, \  
 My\_Preference, Contact, Likes  
from .forms import ContactForm, CustomerForm, VendorForm, NotificationForm, CategoryForm, ProductForm, CartForm, \  
 OrderForm, Order\_ProductsForm, AdminForm, My\_PreferenceForm, ReviewForm, Review, Friends\_Requests, PostForm, Post, \  
 Comment, CommentForm  
from django.http import JsonResponse, HttpResponseForbidden  
  
  
# Create your views here.  
  
def index(request):  
 return render(request, "index.html", {})  
  
  
def doctor(request):  
 return render(request, "doctor.html", {})  
  
  
def departments(request):  
 return render(request, "departments.html", {})  
  
  
def department\_single(request):  
 return render(request, "department-single.html", {})  
  
  
def blog\_single(request):  
 return render(request, "blog-single.html", {})  
  
  
def blog(request):  
 return render(request, "blog.html", {})  
  
  
def about(request):  
 return render(request, "about.html", {})  
  
  
def contact(request):  
 if request.method == "POST":  
 form = ContactForm(request.POST)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, "contact.html", {"msg": "Thanks For Contacting Us"})  
  
 return render(request, "contact.html", {"msg": "While Contacting Us Your Getting Error"})  
 return render(request, "contact.html", {})  
  
  
def customer\_register\_form(request):  
 if request.method == "POST":  
 form = CustomerForm(request.POST, request.FILES)  
 email = request.POST['email']  
 if Customer.objects.filter(email=email).exists():  
 return render(request, "customer\_register\_form.html",  
 {"msg": "This Email Already Exists Please Try With Another Email"})  
  
 else:  
 if form.is\_valid():  
 print(form.errors)  
 form.save()  
 return render(request, "customer\_login.html", {"msg": "Registered Successfully Done"})  
 return render(request, "customer\_register\_form.html", {"msg": "Not Registered Successfully Done"})  
 return render(request, "customer\_register\_form.html", {})  
  
  
def customer\_login(request):  
 if request.method == 'POST':  
 email = request.POST['email']  
 password = request.POST['password']  
 log = Customer.objects.filter(email=email, password=password)  
 if log.exists():  
 if log[0].status == "Accepted":  
 request.session['email'] = email  
 return render(request, "customer\_home.html", {"msg": "Accepted Successfully"})  
 elif log[0].status == "Rejected":  
 return render(request, "customer\_login.html", {"msg": "Rejected login"})  
  
 else:  
 return render(request, "customer\_login.html", {"msg": "Pending"})  
 return render(request, "customer\_login.html", {"msg": "Login Failed"})  
 return render(request, "customer\_login.html", {})  
  
  
def customer\_home(request):  
 return render(request, "customer\_home.html", {})  
  
  
def customer\_logout(request):  
 return redirect('/customer\_login')  
  
  
def customer\_profile(request):  
 email = request.session['email']  
 profile = Customer.objects.get(email=email)  
 return render(request, "customer\_profile.html", {"x": profile})  
  
  
  
  
def admin\_login(request):  
 if request.method == 'POST':  
 email = request.POST['email']  
 password = request.POST['password']  
 log = Admin.objects.filter(email=email, password=password)  
 if log.exists():  
 request.session['email'] = email  
 return render(request, "admin\_home.html", {})  
 return render(request, "admin\_login.html", {"msg": "Login Failed"})  
 return render(request, "admin\_login.html", {})  
  
  
  
  
def admin\_Logout(request):  
 request.session.flush()  
 return redirect('/admin\_login')  
  
  
def admin\_home(request):  
 return render(request, "admin\_home.html", {})  
  
  
def admin\_viewcontact(request):  
 contact = Contact.objects.all()  
 return render(request, "admin\_viewcontact.html", {"contact": contact})  
  
  
def admin\_viewcustomer(request):  
 customer = Customer.objects.all()  
 return render(request, "admin\_viewcustomer.html", {"customer": customer})  
  
  
def accept\_customer(request, id):  
 cus = Customer.objects.get(id=id)  
 cus.status = 'Accepted'  
 cus.save()  
 return redirect('/admin\_viewcustomer')  
  
  
def reject\_customer(request, id):  
 cus = Customer.objects.get(id=id)  
 cus.status = 'Rejected'  
 cus.save()  
 return redirect('/admin\_viewcustomer')  
  
  
def customer\_edit(request):  
 email = request.session['email']  
 cus = Customer.objects.get(email=email)  
 return render(request, 'customer\_edit.html', {'x': cus})  
  
  
def update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 cus = Customer.objects.get(id=id)  
 form = CustomerForm(request.POST, request.FILES, instance=cus)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/customer\_profile')  
 return render(request, "customer\_edit.html", {})  
 return render(request, "customer\_edit.html", {})  
  
  
def is\_login(request):  
 if request.session.\_\_contains\_\_('email'):  
 return True  
 else:  
 return False  
  
  
def customer\_change\_password(request):  
 email = request.session.get("email")  
 if not email:  
 return redirect('/customer\_login')  
  
 if is\_login(request):  
 if request.method == 'POST':  
 password = request.POST['password']  
 new\_password = request.POST['new\_password']  
 if password == new\_password:  
 return render(request, "customer\_change\_password.html",  
 {"msg": "Your Old Password And New Password Are Same.", "email": email})  
 try:  
 user = Customer.objects.get(email=email, password=password)  
 user.password = new\_password  
 user.save()  
 messages.success(request, 'Password Changed Successfully')  
 return redirect('/customer\_login')  
 except Exception as e:  
 print(e)  
 return render(request, "customer\_change\_password.html", {"msg": "Invalid Credentials", "email": email})  
 return render(request, "customer\_change\_password.html", {"email": email})  
 else:  
 return redirect('/customer\_login')  
  
  
def deactivate(request, id):  
 cus = Customer.objects.get(id=id)  
 cus.status = 'pending'  
 cus.save()  
 return redirect('/customer\_login')  
  
  
def admin\_change\_password(request):  
 email = request.session.get("email")  
 if not email:  
 return redirect('/admin\_login')  
  
 if is\_login(request):  
 if request.method == 'POST':  
 password = request.POST['password']  
 new\_password = request.POST['new\_password']  
 if password == new\_password:  
 return render(request, "admin\_change\_password.html",  
 {"msg": "Your Old Password And New Password Are Same.", "email": email})  
 try:  
 user = Admin.objects.get(email=email, password=password)  
 user.password = new\_password  
 user.save()  
 messages.success(request, 'Password Changed Successfully')  
 return redirect('/admin\_login')  
 except Exception as e:  
 print(e)  
 return render(request, "admin\_change\_password.html", {"msg": "Invalid Credentials", "email": email})  
 return render(request, "admin\_change\_password.html", {"email": email})  
 else:  
 return redirect('/admin\_login')  
  
  
def vendor\_register\_form(request):  
 if request.method == "POST":  
 form = VendorForm(request.POST, request.FILES)  
 email = request.POST['email']  
 if Vendor.objects.filter(email=email).exists():  
 return render(request, "vendor\_register\_form.html",  
 {"msg": "This Email Already Exists Please Try With Another Email"})  
  
 else:  
 if form.is\_valid():  
 print(form.errors)  
 form.save()  
 return render(request, "vendor\_login.html", {"msg": "Registration Successfully Done"})  
 return render(request, "vendor\_register\_form.html", {"msg": "Not Registration Successfully Done"})  
 return render(request, "vendor\_register\_form.html", {})  
  
  
def vendor\_login(request):  
 if request.method == 'POST':  
 email = request.POST['email']  
 password = request.POST['password']  
 log = Vendor.objects.filter(email=email, password=password)  
 if log.exists():  
 request.session['email'] = email  
 return render(request, "vendor\_home.html", {"msg": "Accepted Successfully"})  
 return render(request, "vendor\_login.html", {"msg": "Login Failed"})  
 return render(request, "vendor\_login.html", {})  
  
  
def vendor\_home(request):  
 return render(request, "vendor\_home.html", {})  
  
  
def vendor\_logout(request):  
 return redirect('/vendor\_login')  
  
  
def vendor\_profile(request):  
 email = request.session['email']  
 profile = Vendor.objects.get(email=email)  
 return render(request, "vendor\_profile.html", {"x": profile})  
  
  
def accept\_vendor(request, id):  
 vendor = Vendor.objects.get(id=id)  
 vendor.status = 'accepted'  
 vendor.save()  
 return redirect('/view\_vendor')  
  
  
def reject\_vendor(request, id):  
 vendor = Vendor.objects.get(id=id)  
 vendor.status = 'rejected'  
 vendor.save()  
 return redirect('/view\_vendor')  
  
  
def vendor\_edit(request):  
 email = request.session['email']  
 vendor = Vendor.objects.get(email=email)  
 return render(request, 'vendor\_edit.html', {'x': vendor})  
  
  
def vendor\_update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 vendor = Vendor.objects.get(id=id)  
 form = VendorForm(request.POST, request.FILES, instance=vendor)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/vendor\_profile')  
 return render(request, "vendor\_edit.html", {})  
 return render(request, "vendor\_edit.html", {})  
  
  
def vendor\_change\_password(request):  
 email = request.session.get("email")  
 if not email:  
 return redirect('/vendor\_login')  
  
 if is\_login(request):  
 if request.method == 'POST':  
 password = request.POST['password']  
 new\_password = request.POST['new\_password']  
 if password == new\_password:  
 return render(request, "vendor\_change\_password.html",  
 {"msg": "Your Old Password And New Password Are Same.", "email": email})  
 try:  
 user = Vendor.objects.get(email=email, password=password)  
 user.password = new\_password  
 user.save()  
 messages.success(request, 'Password Changed Successfully')  
 return redirect('/vendor\_login')  
 except Exception as e:  
 print(e)  
 return render(request, "vendor\_change\_password.html", {"msg": "Invalid Credentials", "email": email})  
 return render(request, "vendor\_change\_password.html", {"email": email})  
 else:  
 return redirect('/vendor\_login')  
  
  
def deactivate\_vendor(request, id):  
 vendor = Vendor.objects.get(id=id)  
 vendor.status = 'pending'  
 vendor.save()  
 return redirect('/vendor\_login')  
  
  
def vendor\_view\_customer(request):  
 customer = Customer.objects.all()  
 return render(request, "vendor\_view\_customer.html", {"customer": customer})  
  
  
def add\_notification(request):  
 if request.method == 'POST':  
 form = NotificationForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_notification')  
 return render(request, "add\_notification.html", {})  
 return render(request, "add\_notification.html", {})  
  
  
def view\_notification(request):  
 profile = Notification.objects.all()  
 return render(request, "view\_notification.html", {"profile": profile, })  
  
  
def add\_category(request):  
 if request.method == 'POST':  
 form = CategoryForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_category')  
 return render(request, "add\_category.html", {})  
 return render(request, "add\_category.html", {})  
  
  
def view\_category(request):  
 profile = Category.objects.all()  
 return render(request, "view\_category.html", {"profile": profile, })  
  
  
def view\_vendor(request):  
 customer = Vendor.objects.all()  
 return render(request, "view\_vendor.html", {"customer": customer})  
  
  
def delete\_view\_notification(request, id):  
 notify = Notification.objects.get(id=id)  
 notify.delete()  
 return redirect('/add\_notification')  
  
  
def delete\_view\_category(request, id):  
 cat = Category.objects.get(id=id)  
 cat.delete()  
 return redirect('/add\_category')  
  
  
def add\_product(request):  
 email = request.session['email']  
 category = Category.objects.all()  
 if request.method == 'POST':  
 form = ProductForm(request.POST, request.FILES)  
 print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('view\_product')  
 return render(request, "add\_product.html", {"email": email})  
 return render(request, "add\_product.html", {"email": email, "category": category})  
  
  
def view\_product(request):  
 email = request.session['email']  
 profile = Product.objects.filter(email=email)  
 return render(request, "view\_product.html", {"profile": profile})  
  
  
def customer\_view\_product(request):  
 profile = Product.objects.all()  
 return render(request, "customer\_view\_product.html", {"profile": profile})  
  
  
def delete\_view\_product(request, id):  
 cat = Product.objects.get(id=id)  
 cat.delete()  
 return redirect('/add\_product')  
  
  
def product\_edit(request, id):  
 edit = Product.objects.get(id=id)  
 return render(request, 'product\_edit.html', {'x': edit})  
  
  
def product\_update(request):  
 if request.method == 'POST':  
 id = request.POST['id']  
 product = Product.objects.get(id=id)  
 form = ProductForm(request.POST, request.FILES, instance=product)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return redirect('/view\_product')  
 return render(request, "product\_edit.html", {})  
 return render(request, "product\_edit.html", {})  
  
  
def view\_cart(request):  
 email = request.session['email']  
 cate = Cart.objects.filter(customer\_email=email)  
 tq = 0  
 tc = 0  
 td = 0  
 fc = 0  
 for x in cate:  
 tq = tq + int(x.quantity)  
 tc = tc + int(x.cost \* x.quantity)  
 td = td + int(x.discount \* x.quantity)  
 fc = tc - td  
 return render(request, "view\_cart.html", {"cate": cate, "td": td, "tc": tc, "tq": tq, "fc": fc})  
  
  
def delete\_view\_cart(request, id):  
 cart = Cart.objects.get(id=id)  
 cart.delete()  
 return redirect('/view\_cart')  
  
  
def customer\_view\_vendors(request):  
 vendor = Vendor.objects.all()  
 return render(request, "customer\_view\_vendors.html", {"vendor": vendor})  
  
  
def customer\_view\_notification(request):  
 profile = Notification.objects.all()  
 return render(request, "customer\_view\_notification.html", {"profile": profile, })  
  
  
def add\_to\_cart(request, id):  
 email = request.session["email"]  
 customer = Customer.objects.get(email=email)  
 product = Product.objects.get(id=id)  
  
 if request.method == "POST":  
 # Check if the product is already in the cart  
 existing\_cart\_item = Cart.objects.filter(product=product, customer\_email=customer.email).first()  
  
 if existing\_cart\_item:  
 return render(request, "add\_to\_cart.html", {  
 "msg": "This product is already in your cart.",  
 'email': email,  
 'x': product  
 })  
  
 # Check if there are items in the cart from a different vendor  
 cart\_items = Cart.objects.filter(customer\_email=customer.email)  
  
 if cart\_items.exists():  
 existing\_vendor\_email = cart\_items.first().vendor\_email  
  
 if existing\_vendor\_email != product.email:  
 return render(request, "add\_to\_cart.html", {  
 "msg": "You can only select products from one vendor at a time. Please complete your current order before adding products from another vendor.",  
 "customer": customer,  
 "product": product,  
 "email": email  
 })  
  
 # If all checks pass, add the product to the cart  
 form = CartForm(request.POST, request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, "add\_to\_cart.html", {  
 'msg': 'Product successfully added to your cart',  
 'email': email,  
 'x': product  
 })  
 else:  
 print("Form is not valid:", form.errors)  
  
 return render(request, "add\_to\_cart.html", {  
 'email': email,  
 'x': product  
 })  
  
  
# def add\_to\_cart(request):  
# email = request.session['email']  
# customer = Customer.objects.get(email=email)  
# print(customer)  
# email1 = request.session['email']  
# product = Product.objects.filter(email=email1)  
# print("hii")  
# if request.method == "POST":  
# if Cart.objects.filter(product=product, customer\_email=customer.email).exists():  
# return render(request, "add\_to\_cart.html",  
# {"msg": "This Item Already In Cart", "customer": customer, "product": product,  
# "email": email})  
# else:  
# book = CartForm(request.POST)  
# print("errors", book.errors)  
# print('hii')  
# if book.is\_valid():  
# print("error:", book.errors)  
# book.save()  
# return render(request, "add\_to\_cart.html",  
# {"msg": "Added To Cart", "customer": customer, "product": product,  
# "email": email})  
# return render(request, "add\_to\_cart.html",  
# {"customer": customer.id, "product": product, "email": email})  
# return render(request, "add\_to\_cart.html",  
# {"customer": customer.id, "product": product, "email": email})  
  
  
# def customer\_view\_cart(request):  
# email = request.session["email"]  
# cart = Cart.objects.filter(customer\_email=email)  
# tq = 0  
# tc = 0  
# td = 0  
# fc = 0  
# for x in cart:  
# tq = tq + int(x.quantity)  
# tc = tc + int(x.cost \* x.quantity)  
# td = td + int(x.discount \* x.quantity)  
# fc = fc + tc - td  
# return render(request, "view\_cart.html", {"cart": cart, "td": td, "tc": tc, "tq": tq, "fc": fc})  
  
  
# def checkout(request):  
# cart = Cart.objects.all()  
# tq = 0  
# tc = 0  
# td = 0  
# fc = 0  
# customer\_mail = 0  
# vendor\_mail = 0  
# for x in cart:  
# customer\_mail = x.customer\_email  
# vendor\_mail = x.vendor\_email  
# tq = tq + int(x.quantity)  
# tc = tc + int(x.cost \* x.quantity)  
# td = td + int(x.discount \* x.quantity)  
# fc = fc + tc - td  
# if request.method == "POST":  
# print("hello")  
# order\_form = OrderForm(request.POST)  
# print("errors", order\_form.errors)  
# print('hii')  
# if order\_form.is\_valid():  
# print("error:", order\_form.errors)  
# x = order\_form.save()  
# print("x = ", x.id)  
# cart\_1 = Cart.objects.all()  
# for item in cart\_1:  
# Order\_Products.objects.create(order=x, product=item.product, cost=item.cost, quantity=item.quantity,  
# discount=item.discount, total=item.total, title=item.title,  
# customer\_mail=item.customer\_mail,  
# vendor\_mail=item.vendor\_mail)  
#  
# cart\_1.delete()  
# return render(request, "checkout.html",  
# {"msg": "Your Order Is Success", "customer\_mail": customer\_mail, "vmail": vendor\_mail,  
# "td": td, "tc": tc,  
# "tq": tq, "fc": fc})  
# return render(request, "checkout.html",  
# {"customer\_mail": customer\_mail, "vendor\_mail": vendor\_mail, "td": td, "tc": tc, "tq": tq, "fc": fc})  
def checkout(request):  
 print("Checkout called")  
 cart\_items = Cart.objects.filter(customer\_email=request.session["email"])  
  
 tq = sum(item.quantity for item in cart\_items)  
 tc = sum(item.cost \* item.quantity for item in cart\_items)  
 td = sum(item.discount \* item.quantity for item in cart\_items)  
 fc = tc - td  
  
 if request.method == "POST":  
 order\_form = OrderForm(request.POST)  
 print("POST data received:", request.POST)  
  
 if order\_form.is\_valid():  
 order = order\_form.save()  
 print("Order saved with ID:", order.id)  
  
 for item in cart\_items:  
 Order\_Products.objects.create(  
 order=order,  
 product=item.product,  
 title=item.product.title,  
 quantity=item.quantity,  
 cost=item.cost,  
 discount=item.discount,  
 total=item.total,  
 customer\_email=item.customer\_email,  
 vendor\_email=item.vendor\_email  
 )  
  
 cart\_items.delete()  
 return render(request, "checkout.html", {  
 "msg": "Your Order Is Success",  
 "customer\_email": order.customer\_email,  
 "vendor\_email": order.vendor\_email,  
 "td": td,  
 "tc": tc,  
 "tq": tq,  
 "fc": fc  
 })  
 else:  
 print("Form is not valid:", order\_form.errors)  
 else:  
 print("Request method not POST")  
  
 return render(request, "checkout.html", {  
 "customer\_email": cart\_items[0].customer\_email if cart\_items else "",  
 "vendor\_email": cart\_items[0].vendor\_email if cart\_items else "",  
 "td": td,  
 "tc": tc,  
 "tq": tq,  
 "fc": fc  
 })  
  
  
def my\_orders(request):  
 email = request.session['email']  
 orders = Orders.objects.filter(customer\_email=email)  
 return render(request, "my\_orders.html", {"orders": orders})  
  
  
def my\_orders\_products(request, id):  
 orders = Orders.objects.get(id=id)  
 order\_products = Order\_Products.objects.filter(order\_id=orders.id)  
 return render(request, "my\_orders\_products.html", {"order\_products": order\_products})  
  
  
def customer\_view\_products(request, id):  
 order\_products = Order\_Products.objects.filter(order\_id=id)  
 return render(request, "customer\_view\_products.html", {"order\_products": order\_products})  
  
  
def edit\_item(request, id):  
 cart = Cart.objects.get(id=id)  
 return render(request, "edit\_item.html", {"cart": cart})  
  
  
def edit\_item\_update(request):  
 if request.method == "POST":  
 print("error:")  
 id = request.POST["id"]  
 print("hello")  
 user = Cart.objects.get(id=id)  
 users = CartForm(request.POST, instance=user)  
 print("error:", users.errors)  
 if users.is\_valid():  
 print("error:", users.errors)  
 users.save()  
 return redirect("/customer\_view\_cart")  
 return redirect("/customer\_view\_cart")  
  
  
def remove\_item(request, id):  
 cart = Cart.objects.get(id=id)  
 cart.delete()  
 return redirect("/customer\_view\_cart")  
  
  
def customer\_checkout(request):  
 order = Orders.objects.all()  
 return render(request, "customer\_checkout.html", {"order": order})  
  
def cancel\_order(request, id):  
 order = Orders.objects.get(id=id)  
 if request.method == "POST":  
 print("hii")  
 form = OrderForm(request.POST, instance=order)  
 print("hii22")  
 print(form.errors)  
 print("hii3")  
 if form.is\_valid():  
 print(form.errors)  
 order.status = 'Cancelled'  
 form.save()  
 return redirect("/my\_orders")  
 return render(request, "cancel\_order.html", {"order": order})  
  
  
def customer\_cart\_products(request, id):  
 order\_products = Order\_Products.objects.filter(order\_id=id)  
 return render(request, "customer\_cart\_products.html", {"order\_products": order\_products})  
  
  
def accept\_checkout(request, id):  
 accept = Orders.objects.get(id=id)  
 accept.status = 'Accepted'  
 accept.save()  
 return redirect('/vendor\_checkout')  
  
  
def reject\_checkout(request, id):  
 reject = Orders.objects.get(id=id)  
 reject.status = 'Rejected'  
 reject.save()  
 return redirect('/vendor\_checkout')  
  
  
def delivery\_checkout(request, id):  
 delivery = Orders.objects.get(id=id)  
 delivery.status = 'Delivered'  
 delivery.save()  
 return redirect('/vendor\_checkout')  
  
  
def vendor\_cart\_products(request, id):  
 order\_products = Order\_Products.objects.filter(order\_id=id)  
 return render(request, "vendor\_cart\_products.html", {"order\_products": order\_products})  
  
  
def vendor\_checkout(request):  
 order = Orders.objects.all()  
 return render(request, "vendor\_checkout.html", {"order": order})  
  
  
def customer\_orders(request):  
 if request.method == 'POST':  
 form = OrderForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('customer\_view\_cart')  
 return render(request, "customer\_orders.html", {})  
 return render(request, "customer\_orders.html", {})  
  
  
def order\_products(request):  
 if request.method == 'POST':  
 form = Order\_ProductsForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('customer\_orders')  
 return render(request, "order\_products.html", {})  
 return render(request, "order\_products.html", {})  
  
  
def admin(request):  
 if request.method == 'POST':  
 form = AdminForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('admin\_login')  
 return render(request, "admin.html", {})  
 return render(request, "admin.html", {})  
  
  
def add\_preference(request):  
 email = request.session["email"]  
 category = Category.objects.all()  
  
 if request.method == "POST":  
 form = My\_PreferenceForm(request.POST)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, "add\_preference.html", {"msg": "Submitted Preference Form"})  
  
 return render(request, "add\_preference.html", {"msg": "Not Submitted Preference Form"})  
 return render(request, "add\_preference.html", {"email": email, "category": category})  
  
  
def my\_preference(request):  
 email = request.session['email']  
 preferences = My\_Preference.objects.filter(email=email)  
 profile = []  
 for preference in preferences:  
 products = Product.objects.filter(category=preference.category)  
 profile.extend(products)  
  
 return render(request, "my\_preference.html", {"profile": profile})  
  
  
def customer\_view\_cart\_edit(request, id):  
 cate = Cart.objects.get(id=id)  
 return render(request, "customer\_view\_cart\_edit.html", {"cate": cate})  
  
  
def customer\_view\_cart\_update(request):  
 if request.method == "POST":  
 print("error:")  
 id = request.POST["id"]  
 print("hello")  
 user = Cart.objects.get(id=id)  
 users = CartForm(request.POST, instance=user)  
 print("error:", users.errors)  
 if users.is\_valid():  
 print("error:", users.errors)  
 users.save()  
 return redirect('/view\_cart')  
 return render(request, "customer\_view\_cart.html", {"msg": "not update"})  
 return redirect("/customer\_view\_cart")  
  
  
def vendor\_view\_orders(request):  
 email = request.session['email']  
 orders = Orders.objects.filter(vendor\_email=email)  
 return render(request, "vendor\_view\_orders.html", {"orders": orders})  
  
  
def vendor\_view\_orders\_products(request, id):  
 orders = Orders.objects.get(id=id)  
 order\_products = Order\_Products.objects.filter(order\_id=orders.id)  
 return render(request, "vendor\_view\_orders\_products.html", {"order\_products": order\_products})  
  
  
def accept\_order(request, id):  
 orders = Orders.objects.get(id=id)  
 orders.status = 'Accepted'  
 orders.save()  
 return redirect('/vendor\_view\_orders')  
  
  
def reject\_order(request, id):  
 orders = Orders.objects.get(id=id)  
 orders.status = 'Rejected'  
 orders.save()  
 return redirect('/vendor\_view\_orders')  
  
  
def cancel\_order(request, id):  
 orders = Orders.objects.get(id=id)  
 orders.status = 'Cancelled'  
 orders.save()  
 return redirect('//my\_orders')  
  
  
def search(request):  
 queryval = request.POST["profile"]  
 profile = Product.objects.filter(category\_\_contains=queryval)  
 return render(request, "customer\_view\_product.html", {"profile": profile})  
  
  
def all\_users(request):  
 email = request.session['email']  
 customers = Customer.objects.exclude(email=email)  
  
 # Sent and received requests  
 sent\_requests = Friends\_Requests.objects.filter(from\_email=email)  
 received\_requests = Friends\_Requests.objects.filter(to\_email=email)  
  
 # Requested and accepted requests  
 requested\_ids = list(sent\_requests.values\_list('to\_email', flat=True)) + list(  
 received\_requests.values\_list('from\_email', flat=True))  
  
 # Friends list (accepted requests)  
 accepted\_requests = Friends\_Requests.objects.filter(  
 (Q(from\_email=email) | Q(to\_email=email)), status='Accepted'  
 ).values\_list('from\_email', 'to\_email')  
  
 friends\_ids = [req[0] if req[1] == email else req[1] for req in accepted\_requests]  
  
 return render(request, 'all\_users.html', {  
 "customers": customers,  
 "requested\_ids": requested\_ids,  
 "friends\_ids": friends\_ids  
 })  
  
  
def customer\_add\_review(request, id):  
 email = request.session["email"]  
 profile = Product.objects.get(id=id)  
 if request.method == "POST":  
 form = ReviewForm(request.POST)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, "customer\_add\_review.html", {'msg': 'Review Added Successfully '})  
 return render(request, "customer\_add\_review.html", {})  
 return render(request, 'customer\_add\_review.html', {"email": email, "profile": profile})  
  
  
def customer\_view\_review(request, id):  
 profile = Product.objects.get(id=id)  
 reviews = Review.objects.filter(orders\_id=profile.id)  
 return render(request, 'customer\_view\_review.html', {"reviews": reviews})  
  
  
  
def add\_friends\_requests(request, id):  
 if request.method == "POST":  
 email = request.session['email']  
 customer = get\_object\_or\_404(Customer, id=id)  
  
 # Check if request already exists  
 if Friends\_Requests.objects.filter(from\_email=email, to\_email=customer.email).exists():  
 return JsonResponse({"msg": "Request Already Sent"}, status=400)  
  
 # Create 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\_request(request,id):  
 requests = Friends\_Requests.objects.get(id=id)  
 requests.status='Accepted'  
 requests.save()  
 return redirect("/view\_friends\_requests")  
  
def reject\_request(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']  
 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 post(request):  
 email = request.session['email']  
 if request.method == "POST":  
 form = PostForm(request.POST,request.FILES)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, "post.html", {"msg": "Thanks For Posting Us"})  
 return render(request, "post.html", {"msg": "While Posting Us Your Getting Error"})  
 return render(request, "post.html", {"email":email})  
  
  
  
  
  
def view\_post(request):  
 email = request.session['email']  
 friends\_pairs = Friends\_Requests.objects.filter(  
 (Q(from\_email=email) | Q(to\_email=email)),  
 status='Accepted'  
 ).values\_list('from\_email','to\_email')  
  
 friends = set(chain(\*friends\_pairs)) - {email}  
  
 post = Post.objects.filter(  
 Q(posts\_for='all') |  
 (Q(posts\_for = 'friends') & Q(email\_\_in=friends)),  
 status='Accepted'  
 ).exclude(email=email).annotate(like\_count=Count('likes'))  
  
 liked\_posts = Likes.objects.filter(customer\_email=email).values\_list('post\_id',flat=True)  
 return render(request, "view\_post.html", {"post":post,"liked\_posts":liked\_posts})  
  
  
def view\_all\_posts(request):  
 email = request.session['email']  
 post = Post.objects.filter(email=email)  
 return render(request, "view\_all\_posts.html", {"post":post})  
  
def admin\_view\_all\_posts(request):  
 post = Post.objects.all()  
 return render(request, "admin\_view\_all\_posts.html", {"post":post})  
  
  
def accept\_all\_posts(request,id):  
 requests = Post.objects.get(id=id)  
 requests.status='Accepted'  
 requests.save()  
 return redirect("/admin\_view\_all\_posts")  
  
  
  
def reject\_all\_posts(request, id):  
 requests = Post.objects.get(id=id)  
 requests.status='Rejected'  
 requests.save()  
 return redirect("/admin\_view\_all\_posts")  
  
def like\_post(request, post\_id):  
 email = request.session['email']  
 post = get\_object\_or\_404(Post, id=post\_id)  
  
 if not Likes.objects.filter(post=post, customer\_email=email).exists():  
 Likes.objects.create(post=post, customer\_email=email, date\_time=timezone.now())  
 msg = "Liked Successfully"  
 else:  
 msg = "You've Already Likes This Post"  
 return redirect('/view\_post')  
  
  
  
def comment(request, id):  
 email = request.session["email"]  
 profile = Post.objects.get(id=id)  
 comments = Comment.objects.filter(post\_id\_id=profile.id)  
 if request.method == "POST":  
 form = CommentForm(request.POST)  
 print(form.errors)  
 if form.is\_valid():  
 form.save()  
 return render(request, "comment.html", {'msg': 'Comment Added Successfully ',"comments":comments})  
 return render(request, "comment.html", {"comments":comments})  
 return render(request, 'comment.html', {"email": email, "profile": profile,"comments":comments})

URLS:

from django.conf import settings  
from django.conf.urls.static import static  
from django.contrib import admin  
from django.urls import path  
from pythonfratelloapp import views  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('',views.index,name="index"),  
 path('doctor',views.doctor,name="doctor"),  
 path('contact', views.contact, name="contact"),  
 path('departments', views.departments, name="departments"),  
 path('department', views.department\_single, name="department"),  
 path('blog', views.blog, name="blog"),  
 path('blogs', views.blog\_single, name="blogs"),  
 path('about', views.about, name="about"),  
 path('customer\_login',views.customer\_login, name="customer\_login"),  
 path('customer\_register\_form',views.customer\_register\_form,name="customer\_register\_form"),  
 path('customer\_home',views.customer\_home,name="customer\_home"),  
 path('customer\_profile',views.customer\_profile,name="customer\_profile"),  
 path('customer\_logout',views.customer\_logout,name="customer\_logout"),  
 path('admin\_login',views.admin\_login,name="admin\_login"),  
 path('admin\_home',views.admin\_home,name="admin\_home"),  
 path('admin\_Logout',views.admin\_Logout,name="admin\_Logout"),  
 path('admin\_viewcontact',views.admin\_viewcontact,name="admin\_viewcontact"),  
 path('admin\_viewcustomer',views.admin\_viewcustomer,name="admin\_viewcustomer"),  
 path('accept\_customer/<int:id>',views.accept\_customer,name="accept\_customer"),  
 path('reject\_customer/<int:id>', views.reject\_customer, name="reject\_customer"),  
 path('customer\_edit',views.customer\_edit,name="customer\_edit"),  
 path('update',views.update,name="update"),  
 path('customer\_change\_password',views.customer\_change\_password,name="customer\_change\_password"),  
 path('deactivate/<int:id>', views.deactivate, name="deactivate"),  
 path('admin\_change\_password', views.admin\_change\_password, name="admin\_change\_password"),  
 path('vendor\_login', views.vendor\_login, name="vendor\_login"),  
 path('vendor\_register\_form', views.vendor\_register\_form, name="vendor\_register\_form"),  
 path('vendor\_home', views.vendor\_home, name="vendor\_home"),  
 path('vendor\_profile', views.vendor\_profile, name="vendor\_profile"),  
 path('vendor\_logout', views.vendor\_logout, name="vendor\_logout"),  
 path('vendor\_view\_customer',views.vendor\_view\_customer,name="vendor\_view\_customer"),  
 path('accept\_vendor/<int:id>',views.accept\_vendor,name="accept\_vendor"),  
 path('reject\_vendor/<int:id>', views.reject\_vendor, name="reject\_vendor"),  
 path('vendor\_edit',views.vendor\_edit,name="vendor\_edit"),  
 path('vendor\_update',views.vendor\_update,name="vendor\_update"),  
 path('vendor\_change\_password',views.vendor\_change\_password,name="vendor\_change\_password"),  
 path('deactivate\_vendor/<int:id>', views.deactivate\_vendor, name="deactivate\_vendor"),  
 path('add\_notification',views.add\_notification,name="add\_notification"),  
 path('view\_notification', views.view\_notification, name="view\_notification"),  
 path('add\_category/', views.add\_category, name="add\_category"),  
 path('view\_category', views.view\_category, name="view\_category"),  
 path('view\_vendor', views.view\_vendor, name="view\_vendor"),  
 path('delete\_view\_notification/<int:id>',views.delete\_view\_notification, name="delete\_view\_notification"),  
 path('delete\_view\_category/<int:id>', views.delete\_view\_category, name="delete\_view\_category"),  
 path('add\_product', views.add\_product, name="add\_product"),  
 path('view\_product', views.view\_product, name="view\_product"),  
 path('customer\_view\_product', views.customer\_view\_product, name="customer\_view\_product"),  
 path('delete\_view\_product/<int:id>', views.delete\_view\_product, name="delete\_view\_product"),  
 path('product\_edit/<int:id>', views.product\_edit, name="product\_edit"),  
 path('product\_update', views.product\_update, name="product\_update"),  
 path('add\_to\_cart/<int:id>', views.add\_to\_cart, name="add\_to\_cart"),  
 path('view\_cart', views.view\_cart, name="view\_cart"),  
 path('delete\_view\_cart/<int:id>', views.delete\_view\_cart, name="delete\_view\_cart"),  
 path('customer\_view\_vendors', views.customer\_view\_vendors, name="customer\_view\_vendors"),  
 path('customer\_view\_notification', views.customer\_view\_notification, name="customer\_view\_notification"),  
 # path('customer\_view\_cart', views.customer\_view\_cart, name="customer\_view\_cart"),  
 path('checkout', views.checkout, name="checkout"),  
 path('edit\_item', views.edit\_item, name="edit\_item"),  
 path('edit\_item\_update', views.edit\_item\_update, name="edit\_item\_update"),  
 path('remove\_item', views.remove\_item, name="remove\_item"),  
 path('cancel\_order', views.cancel\_order, name="cancel\_order"),  
 path('customer\_checkout', views.customer\_checkout, name="customer\_checkout"),  
 path('customer\_cart\_products', views.customer\_cart\_products, name="customer\_cart\_products"),  
 path('accept\_checkout', views.accept\_checkout, name="accept\_checkout"),  
 path('reject\_checkout', views.reject\_checkout, name="reject\_checkout"),  
 path('delivery\_checkout', views.delivery\_checkout, name="delivery\_checkout"),  
 path('vendor\_cart\_products', views.vendor\_cart\_products, name="vendor\_cart\_products"),  
 path('vendor\_checkout', views.vendor\_checkout, name="vendor\_checkout"),  
 path('customer\_orders', views.customer\_orders, name="customer\_orders"),  
 path('order\_products', views.order\_products, name="order\_products"),  
 path('admin', views.admin, name="admin"),  
 path('add\_preference', views.add\_preference, name="add\_preference"),  
 path('my\_preference', views.my\_preference, name="my\_preference"),  
 path('customer\_view\_cart\_edit/<int:id>', views.customer\_view\_cart\_edit, name="customer\_view\_cart\_edit"),  
 path('customer\_view\_cart\_update', views.customer\_view\_cart\_update, name="customer\_view\_cart\_update"),  
 path('my\_orders', views.my\_orders, name="my\_orders"),  
 path('my\_orders\_products/<int:id>', views.my\_orders\_products, name="my\_orders\_products"),  
 path('vendor\_view\_orders', views.vendor\_view\_orders, name="vendor\_view\_orders"),  
 path('vendor\_view\_orders\_products/<int:id>', views.vendor\_view\_orders\_products, name="vendor\_view\_orders\_products"),  
 path('accept\_order/<int:id>', views.accept\_order, name="accept\_order"),  
 path('reject\_order/<int:id>', views.reject\_order, name="reject\_order"),  
 path('cancel\_order/<int:id>', views.cancel\_order, name="cancel\_order"),  
 path('search', views.search, name="search"),  
 path('all\_users', views.all\_users, name="all\_users"),  
 path('customer\_add\_review/<int:id>', views.customer\_add\_review, name="customer\_add\_review"),  
 path('customer\_view\_review/<int:id>', views.customer\_view\_review, name="customer\_view\_review"),  
 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\_request/<int:id>', views.accept\_request, name="accept\_request"),  
 path('reject\_request/<int:id>', views. reject\_request, name="reject\_request"),  
 path('my\_friends', views.my\_friends, name="my\_friends"),  
 path('post', views.post, name="post"),  
 path('view\_post', views.view\_post, name="view\_post"),  
 path('view\_all\_posts', views.view\_all\_posts, name="view\_all\_posts"),  
 path('admin\_view\_all\_posts', views.admin\_view\_all\_posts, name="admin\_view\_all\_posts"),  
 path('accept\_all\_posts/<int:id>', views.accept\_all\_posts, name="accept\_all\_posts"),  
 path('reject\_all\_posts/<int:id>', views.reject\_all\_posts, name="reject\_all\_posts"),  
  
 path('like\_post/<int:post\_id>', views.like\_post, name='like\_post'),  
 path('comment/<int:id>', views.comment, name="comment")  
  
]  
  
  
if settings.DEBUG: # new  
 urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

FORMS:

from django import forms  
from .models import Contact,Customer,Vendor,Notification,Category,Product,Cart,Orders,Order\_Products,Admin, My\_Preference,Review,Friends\_Requests,Post,Likes,Comment  
class ContactForm(forms.ModelForm):  
 class Meta:  
 model = Contact  
 fields = "\_\_all\_\_"  
class CustomerForm(forms.ModelForm):  
 class Meta:  
 model = Customer  
 fields = "\_\_all\_\_"  
 exclude=['status']  
class VendorForm(forms.ModelForm):  
 class Meta:  
 model = Vendor  
 fields = "\_\_all\_\_"  
 exclude=['status']  
class NotificationForm(forms.ModelForm):  
 class Meta:  
 model = Notification  
 fields = "\_\_all\_\_"  
class CategoryForm(forms.ModelForm):  
 class Meta:  
 model = Category  
 fields= "\_\_all\_\_"  
class ProductForm(forms.ModelForm):  
 class Meta:  
 model = Product  
 fields = "\_\_all\_\_"  
class CartForm(forms.ModelForm):  
 class Meta:  
 model = Cart  
 fields = "\_\_all\_\_"  
class OrderForm(forms.ModelForm):  
 class Meta:  
 model = Orders  
 exclude = ["status"]  
class Order\_ProductsForm(forms.ModelForm):  
 class Meta:  
 model = Order\_Products  
 fields = "\_\_all\_\_"  
class AdminForm(forms.ModelForm):  
 class Meta:  
 model = Admin  
 fields = "\_\_all\_\_"  
  
  
class My\_PreferenceForm(forms.ModelForm):  
 class Meta:  
 model = My\_Preference  
 fields = "\_\_all\_\_"  
  
  
class ReviewForm(forms.ModelForm):  
 class Meta:  
 model =Review  
 fields = "\_\_all\_\_"  
  
  
class Friends\_RequestsForm(forms.ModelForm):  
 class Meta:  
 model =Friends\_Requests  
 exclude = ["status"]  
  
  
  
class PostForm(forms.ModelForm):  
 class Meta:  
 model = Post  
 exclude = ["status"]  
  
  
class LikesForm(forms.ModelForm):  
 class Meta:  
 model = Likes  
 fields = "\_\_all\_\_"  
  
class CommentForm(forms.ModelForm):  
 class Meta:  
 model = Comment  
 fields = "\_\_all\_\_"

MODELS:

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=50)  
 email = models.EmailField()  
 subject = models.CharField(max\_length=40)  
 message = models.TextField()  
  
 class Meta:  
 db\_table = "Contact"  
  
  
class Admin(models.Model):  
 email = models.EmailField()  
 password = models.CharField(max\_length=100)  
  
  
class Customer(models.Model):  
 name = models.CharField(max\_length=60)  
 email = models.EmailField()  
 mandal = models.CharField(max\_length=50)  
 district = models.CharField(max\_length=50)  
 state = models.CharField(max\_length=40)  
 age = models.IntegerField()  
 gender = models.CharField(max\_length=40)  
 image = models.FileField()  
 password = models.CharField(max\_length=20)  
 status = models.CharField(max\_length=100, default='pending')  
  
  
class Vendor(models.Model):  
 name = models.CharField(max\_length=60)  
 email = models.EmailField()  
 mandal = models.CharField(max\_length=50)  
 district = models.CharField(max\_length=50)  
 state = models.CharField(max\_length=40)  
 age = models.IntegerField()  
 gender = models.CharField(max\_length=40)  
 image = models.FileField()  
 password = models.CharField(max\_length=20)  
 status = models.CharField(max\_length=100, default='pending')  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class Notification(models.Model):  
 title = models.CharField(max\_length=100)  
 description = models.TextField()  
 date\_time = models.DateTimeField(auto\_now=True)  
  
  
class Category(models.Model):  
 title = models.CharField(max\_length=100)  
  
  
class Product(models.Model):  
 email = models.EmailField()  
 category = models.CharField(max\_length=100)  
 title = models.CharField(max\_length=40)  
 orign = models.CharField(max\_length=80)  
 price = models.DecimalField(max\_digits=10, decimal\_places=2)  
 image = models.FileField()  
 description = models.TextField()  
 brand = models.CharField(max\_length=30)  
 model = models.CharField(max\_length=30)  
 discount = models.IntegerField()  
 specification = models.CharField(max\_length=40)  
 warranty = models.CharField(max\_length=40)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class Cart(models.Model):  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 title = models.CharField(max\_length=100)  
 quantity = models.BigIntegerField()  
 cost = models.BigIntegerField()  
 discount = models.BigIntegerField()  
 total = models.BigIntegerField()  
 customer\_email = models.EmailField()  
 vendor\_email = models.EmailField()  
  
 class Meta:  
 db\_table = 'Cart'  
  
  
class Orders(models.Model):  
 customer\_email = models.EmailField()  
 vendor\_email = models.EmailField()  
 quantity = models.BigIntegerField()  
 cost = models.BigIntegerField()  
 discount = models.BigIntegerField()  
 total = models.BigIntegerField()  
 address = models.TextField()  
 delivery\_date = models.DateField()  
 order\_datetime = models.DateTimeField(auto\_now=True)  
 status = models.CharField(max\_length=100, default="Pending")  
  
  
class Order\_Products(models.Model):  
 order = models.ForeignKey(Orders, on\_delete=models.CASCADE)  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 title = models.CharField(max\_length=100)  
 quantity = models.BigIntegerField()  
 cost = models.BigIntegerField()  
 discount = models.BigIntegerField()  
 total = models.BigIntegerField()  
 customer\_email = models.EmailField()  
 vendor\_email = models.EmailField()  
  
  
class My\_Preference(models.Model):  
 email = models.EmailField()  
 category = models.CharField(max\_length=100)  
  
  
class Review(models.Model):  
 customer\_email = models.EmailField()  
 orders = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 reviews = models.CharField(max\_length=100)  
 date\_time = models.DateTimeField(auto\_now=True)  
 ratings = models.BigIntegerField()  
  
 class Meta:  
 db\_table = "Review"  
  
  
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 Post(models.Model):  
 posts\_for = models.CharField(max\_length=100)  
 email = models.EmailField()  
 title = models.CharField(max\_length=100)  
 description = models.TextField()  
 date\_time = models.DateTimeField(auto\_now=True)  
 file = models.FileField()  
 status = models.CharField(max\_length=100, default="Pending")  
  
 class Meta:  
 db\_table = "Post"  
  
  
class Likes(models.Model):  
 post = models.ForeignKey(Post, on\_delete=models.CASCADE)  
 customer\_email = models.EmailField()  
 date\_time = models.DateTimeField(default=timezone.now)  
  
 class Meta:  
 unique\_together = ('post', 'customer\_email')  
  
  
class Review(models.Model):  
 customer\_email = models.EmailField()  
 orders = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 reviews = models.CharField(max\_length=100)  
 date\_time = models.DateTimeField(auto\_now=True)  
 ratings = models.BigIntegerField()  
  
 class Meta:  
 db\_table = "Review"  
  
  
class Comment(models.Model):  
 customer\_email = models.EmailField()  
 post\_id = models.ForeignKey(Post,on\_delete=models.CASCADE)  
 comment = models.CharField(max\_length=100)  
 date\_time = models.DateTimeField(auto\_now=True)  
  
 class Meta:  
 db\_table = "Comment"

CUSTOMER LOGIN:

{% load static %}  
{% include "header.html" %}  
   
 <div class="hero-wrap" style="background-image: url({% static 'images/S8.jpg' %}); background-attachment:fixed;">  
 <div class="overlay"></div>  
 <div class="container">  
 <div class="row no-gutters slider-text align-items-center justify-content-center" data-scrollax-parent="true">  
 <div class="col-md-8 ftco-animate text-center">  
 <h1 class="mb-3 bread"> Login Here</h1>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <section class="ftco-section contact-section ftco-degree-bg">  
 <div class="container">  
 <div class="row d-flex mb-5 contact-info">  
 <div class="col-md-12 mb-4">  
  
  
 <div class="row block-9">  
 <div class="col-md-6 pr-md-5">  
 <form method="post">  
 {% csrf\_token %}  
  
 <div class="form-group">  
 <input type="email" class="form-control" placeholder="Your Email" name="email" required>  
 </div>  
 <div class="form-group">  
 <input type="password" class="form-control" placeholder="Your Password" name="password" required>  
 </div>  
  
 <div class="form-group">  
 <input type="submit" value="Login" class="btn btn-primary py-3 px-5" placeholder="Login" required>  
 </div>  
 <p><b>NOTE:If You Are Not Registered?</b></p>  
 <a href="{% url 'customer\_register\_form' %}">Register Here</a>  
<b style="color:red">  
 {{msg}}  
 {% if messages %}  
 {% for message in messages %}  
 {{ message }}  
 {% endfor %}  
 {% endif%}  
 </b>  
 </form>  
   
 </div>  
  
 </div>  
 </div>  
  
 </section>  
   
  
  
 {% include "footer.html" %}