from django.shortcuts import render, redirect

from django.http import JsonResponse

import json

import datetime

from django.http import HttpResponse

from django.forms import inlineformset\_factory

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth import authenticate, login, logout

from django.contrib import messages

from django.contrib.auth.decorators import login\_required

from django.contrib.auth.models import Group

from django.contrib.auth.models import User

# Create your views here.

from .models import \*

from .forms import OrderItemForm, CreateUserForm, CustomerForm, CommentForm

from .filters import ProductFilter, OrderFilter, OrderMatchFilter, DeliveryFilter

from .decorators import unauthenticated\_user, allowed\_users, admin\_only

def home(request):

context = {}

return render(request, 'store/home.html', context)

@login\_required(login\_url='login')

@admin\_only

def homein(request):

context = {}

return render(request, 'store/homein.html', context)

def store(request):

products = Product.objects.all()

myFilter = ProductFilter(request.GET, queryset=products)

products = myFilter.qs

context = {'products': products, 'myFilter': myFilter}

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

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin', 'customer'])

def storein(request):

products = Product.objects.all()

myFilter = ProductFilter(request.GET, queryset=products)

products = myFilter.qs

context = {'products': products, 'myFilter': myFilter}

return render(request, 'store/storein.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['customer'])

def cart(request):

if request.user.is\_authenticated:

customer = request.user.customer

order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

items = order.orderitem\_set.all()

cartItems = order.get\_cart\_items

else:

items = []

order = {'get\_cart\_total': 0, 'get\_cart\_items': 0, 'shipping': False}

cartItems = order['get\_cart\_items']

context = {'items': items, 'order': order, 'cartItems': cartItems}

return render(request, 'store/cart.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['customer'])

def checkout(request):

if request.user.is\_authenticated:

customer = request.user.customer

order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

items = order.orderitem\_set.all()

cartItems = order.get\_cart\_items

else:

items = []

order = {'get\_cart\_total': 0, 'get\_cart\_items': 0, 'shipping': False}

cartItems = order['get\_cart\_items']

context = {'items': items, 'order': order, 'cartItems': cartItems}

return render(request, 'store/checkout.html', context)

def comments(request):

context = {}

return render(request, 'store/comments.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin', 'customer'])

def commentsin(request):

comments = Comment.objects.all()

context = {'comments': comments}

return render(request, 'store/commentsin.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin', 'customer'])

def writeComment(request):

writer = request.user

form = CommentForm(instance=writer)

if request.method == 'POST':

form = CommentForm(request.POST, request.FILES, instance=writer)

if form.is\_valid():

form.save()

context = {'form': form,'writer': writer}

return render(request, 'store/write\_comment.html', context)

def contact(request):

context = {}

return render(request, 'store/contact.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin', 'customer'])

def contactin(request):

context = {}

return render(request, 'store/contactin.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['customer'])

def userpage(request):

orders = request.user.customer.order\_set.all()

print('ORDERS:', orders)

context = {'orders': orders}

return render(request, 'store/customer.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['customer'])

def customersetting(request):

customer = request.user.customer

form = CustomerForm(instance=customer)

if request.method == 'POST':

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

if form.is\_valid():

form.save()

context = {'form': form, 'customer': customer}

return render(request, 'store/customer\_setting.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['customer'])

def order(request, pk\_test):

order = Order.objects.get(id=pk\_test)

items = order.orderitem\_set.all()

ads = order.shippingaddress\_set.all()

context = {'order': order, 'items': items, 'ads': ads}

return render(request, 'store/order.html', context)

@unauthenticated\_user

def loginpage(request):

if request.method == 'POST':

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

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

user = authenticate(request, username=username, password=password)

if user is not None:

login(request, user)

return redirect('homein')

else:

messages.info(request, 'Username OR password is incorrect')

context = {}

return render(request, 'store/login.html', context)

def logoutuser(request):

logout(request)

return redirect('home')

@unauthenticated\_user

def register(request):

form = CreateUserForm()

if request.method == 'POST':

form = CreateUserForm(request.POST)

if form.is\_valid():

user = form.save()

username = form.cleaned\_data.get('username')

group = Group.objects.get(name='customer')

user.groups.add(group)

Customer.objects.create(

user=user,

name=user.username,

)

messages.success(request, 'Account was created for ' + username)

return redirect('login')

context = {'form': form}

return render(request, 'store/register.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def dashboard(request):

items = OrderItem.objects.select\_related('product', 'order')

myFilter = OrderFilter(request.GET, queryset=items)

items = myFilter.qs

context = { 'items': items, 'myFilter': myFilter }

return render(request, 'store/dashboard.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def order\_match(request):

orders = Order.objects.all()

myFilter = OrderMatchFilter(request.GET, queryset=orders)

orders = myFilter.qs

context = {'orders': orders, 'myFilter': myFilter }

return render(request, 'store/order\_match.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def reports(request):

context = {}

return render(request, 'store/reports.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def delivery(request):

ads = ShippingAddress.objects.select\_related('order','customer')

myFilter = DeliveryFilter(request.GET, queryset=ads)

ads = myFilter.qs

context = {'ads': ads, 'myFilter': myFilter }

return render(request, 'store/delivery.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def updateorder(request, pk):

orderitem = OrderItem.objects.get(id=pk)

form = OrderItemForm(instance=orderitem)

if request.method == 'POST':

form = OrderItemForm(request.POST, instance=orderitem)

if form.is\_valid():

form.save()

return redirect('/dashboard/')

context = {'form': form}

return render(request, 'store/order\_form.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def deleteorder(request, pk):

orderitem = OrderItem.objects.get(id=pk)

if request.method == "POST":

orderitem.delete()

return redirect('/dashboard/')

context = {'item': orderitem}

return render(request, 'store/delete.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['customer'])

def updateItem(request):

data = json.loads(request.body)

productId = data['productId']

action = data['action']

print('Action:', action)

print('Product:', productId)

customer = request.user.customer

product = Product.objects.get(id=productId)

order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

orderItem, created = OrderItem.objects.get\_or\_create(order=order, product=product)

if action == 'add':

orderItem.quantity = (orderItem.quantity + 1)

elif action == 'remove':

orderItem.quantity = (orderItem.quantity - 1)

orderItem.save()

if orderItem.quantity <= 0:

orderItem.delete()

return JsonResponse('Item was added', safe=False)

def processOrder(request):

transaction\_id = datetime.datetime.now().timestamp()

data = json.loads(request.body)

if request.user.is\_authenticated:

customer = request.user.customer

order, created = Order.objects.get\_or\_create(customer=customer, complete=False)

total = float(data['form']['total'])

order.transaction\_id = transaction\_id

if total == order.get\_cart\_total:

order.complete = True

order.save()

if order.shipping == True:

ShippingAddress.objects.create(

customer=customer,

order=order,

address=data['shipping']['address'],

city=data['shipping']['city'],

state=data['shipping']['state'],

zipcode=data['shipping']['zipcode'],

)

else:

print('User is not logged in')

return JsonResponse('Payment submitted..', safe=False)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def reports(request):

websitereports = WebsiteReport.objects.all()

farmreports = FarmReport.objects.all()

overallreports = OverallReport.objects.all()

context = {'websitereports': websitereports, 'farmreports': farmreports, 'overallreports': overallreports}

return render(request, 'store/reports.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def websiteReport(request, pk):

websitereport = WebsiteReport.objects.get(id=pk)

context = {'websitereport': websitereport}

return render(request, 'store/website\_report.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def farmReport(request, pk):

farmreport = FarmReport.objects.get(id=pk)

context = {'farmreport': farmreport}

return render(request, 'store/farm\_report.html', context)

@login\_required(login\_url='login')

@allowed\_users(allowed\_roles=['admin'])

def overallReport(request, pk):

overallreport = OverallReport.objects.get(id=pk)

context = {'overallreport': overallreport}

return render(request, 'store/overall\_report.html', context)