

# Account

## admin.py

codeinstitute

CI Python Linter

```
1 from django.contrib import admin
2 from .models import Customer
3
4
5 class UserBaseAdmin(admin.ModelAdmin):
6     """
7     Customizes the admin interface for user base model.
8     """
9     list_display = ('user_name', 'email', 'profile_image_thumbnail')
10
11 def profile_image_thumbnail(self, obj):
12     """
13     Renders a thumbnail of the user's profile image.
14     """
15     if obj.profile_image:
16         return ('' %
17                 obj.profile_image.url)
18     else:
19         return 'No image'
20 profile_image_thumbnail.allow_tags = True
21
22
23 admin.site.register(Customer)
24
```

Settings:

Results:

All clear, no errors found

## apps.py

codeinstitute

CI Python Linter

```
1 from django.apps import AppConfig
2
3
4 class AccountConfig(AppConfig):
5     """
6     Configures the account app.
7     """
8     default_auto_field = 'django.db.models.BigAutoField'
9     name = 'account'
10
```

Settings:

Results:

All clear, no errors found

# Account

## forms.py

codeinstitute

CI Python Linter

```
1 from django import forms
2 from django.contrib.auth.forms import (AuthenticationForm, PasswordResetForm,
3   SetPasswordForm)
4
5 from .models import Customer, Address
6
7
8 class UserAddressForm(forms.ModelForm):
9     """
10     Form for user address information.
11     """
12     class Meta:
13         model = Address
14         fields = [
15             "full_name",
16             "phone",
17             "address_line",
18             "address_line2",
19             "town_city",
20             "postcode",
21             "country"
22         ]
23
24 def __init__(self, *args, **kwargs):
25     """
26     Initialize form fields and attributes.
27     """
28     super().__init__(*args, **kwargs)
29     self.fields["full_name"].widget.attrs.update(
30         {
31             "class": "form-control mb-2 account-form",
32             "placeholder": "Full Name"
33         })
```

Settings:

Results:

All clear, no errors found

## models.py

codeinstitute

CI Python Linter

```
1 import uuid
2 from django.db import models
3 from django.core.mail import send_mail
4 from django.utils.translation import gettext_lazy as _
5 from django.contrib.auth.models import (AbstractBaseUser, BaseUserManager,
6   PermissionsMixin)
7 from django_countries.fields import CountryField
8
9
10 class CustomAccountManager(BaseUserManager):
11     """
12     Custom user manager for handling user creation.
13     """
14
15     def create_superuser(self, email, name, password, **other_fields):
16         """
17         Creates and saves a superuser with the given email, name, and password.
18         """
19         other_fields.setdefault('is_staff', True)
20         other_fields.setdefault('is_superuser', True)
21         other_fields.setdefault('is_active', True)
22
23         if other_fields.get('is_staff') is not True:
24             raise ValueError(
25                 'Superuser must be assigned to is_staff=True.')
26         if other_fields.get('is_superuser') is not True:
27             raise ValueError(
28                 'Superuser must be assigned to is_superuser=True.')
29
30         return self.create_user(email, name, password, **other_fields)
```

Settings:

Results:

All clear, no errors found

# Account

## tokens.py

codeinstitute

CI Python Linter

```
1 from django.contrib.auth.tokens import PasswordResetTokenGenerator
2 from six import text_type
3
4
5 class AccountActivationTokenGenerator(PasswordResetTokenGenerator):
6     """
7     Token generator for account activation.
8     """
9     def _make_hash_value(self, user, timestamp):
10         """
11         Generates a hash value for the token.
12         """
13         return (
14             text_type(user.pk) + text_type(timestamp) +
15             text_type(user.is_active)
16         )
17
18
19 account_activation_token = AccountActivationTokenGenerator()
20
```

Settings:

☐ ☒ ☐

Results:

All clear, no errors found

## urls.py

codeinstitute

CI Python Linter

```
1 """
2 URL configuration for account app.
3
4 """
5
6 from django.contrib.auth import views as auth_views
7 from django.urls import path
8 from django.views.generic import TemplateView
9
10 from . import views
11 from .views import custom_logout
12 from .forms import (PwdResetConfirmForm, PwdResetForm, UserLoginForm)
13
14
15 app_name = 'account'
16
17 urlpatterns = [
18     path('login/', auth_views.LoginView.as_view(
19         template_name='account/registration/login.html',
20         form_class=UserLoginForm), name='login'),
21     path('logout/', custom_logout, name='logout'),
22     path('register/', views.account_register, name='register'),
23     path(
24         'activate/<slug:uidb64>/<slug:token>/',
25         views.account_activate,
26         name='activate'
27     ),
28     # Reset password
29     path('password_reset/', auth_views.PasswordResetView.as_view(
30         template_name="account/user/password_reset_form.html",
31         success_url='password_reset_email_confirm',
32
```

Settings:

☐ ☒ ☐

Results:

All clear, no errors found

# Account

## views.py



### CI Python Linter

```
1 from django.contrib import messages
2 from django.contrib.auth import login, logout
3 from django.contrib.auth.decorators import login_required
4 from django.contrib.sites.shortcuts import get_current_site
5 from django.http import HttpResponse, HttpResponseRedirect, JsonResponse
6 from django.shortcuts import get_object_or_404, redirect, render
7 from django.urls import reverse
8 from django.template.loader import render_to_string
9 from django.utils.encoding import force_bytes, force_str
10 from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode
11
12 from orders.views import user_orders
13 from store.models import Product
14 from .forms import RegistrationForm, UserEditForm, UserAddressForm
15 from .models import Customer, Address
16 from .tokens import account_activation_token
17 from newsletter.views import check_subscription_status
18
19
20 @login_required
21 def wishlist(request):
22     """
23     View to display user's wishlist.
24     """
25     products = Product.objects.filter(users_wishlist=request.user)
26     return render(
27         request,
28         "account/user/user_wish_list.html",
29         {"wishlist": products}
30     )
31
32
```

#### Settings:



#### Results:

All clear, no errors found