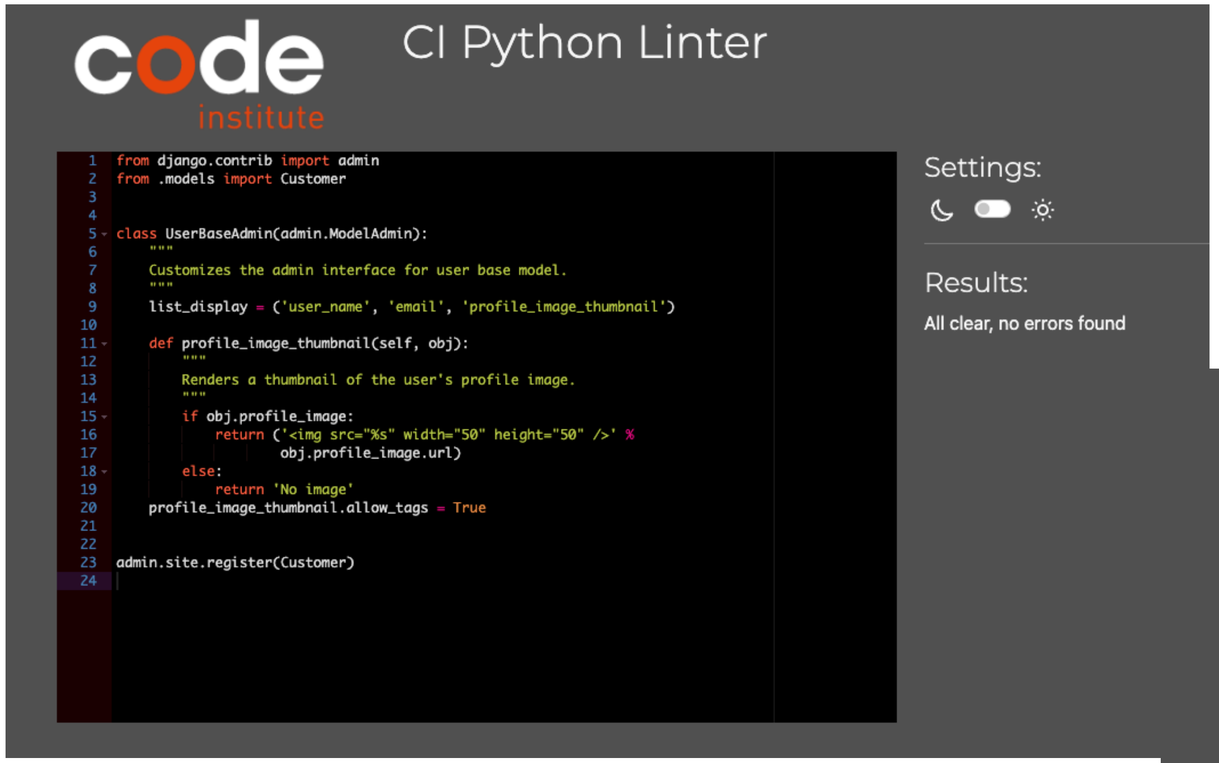
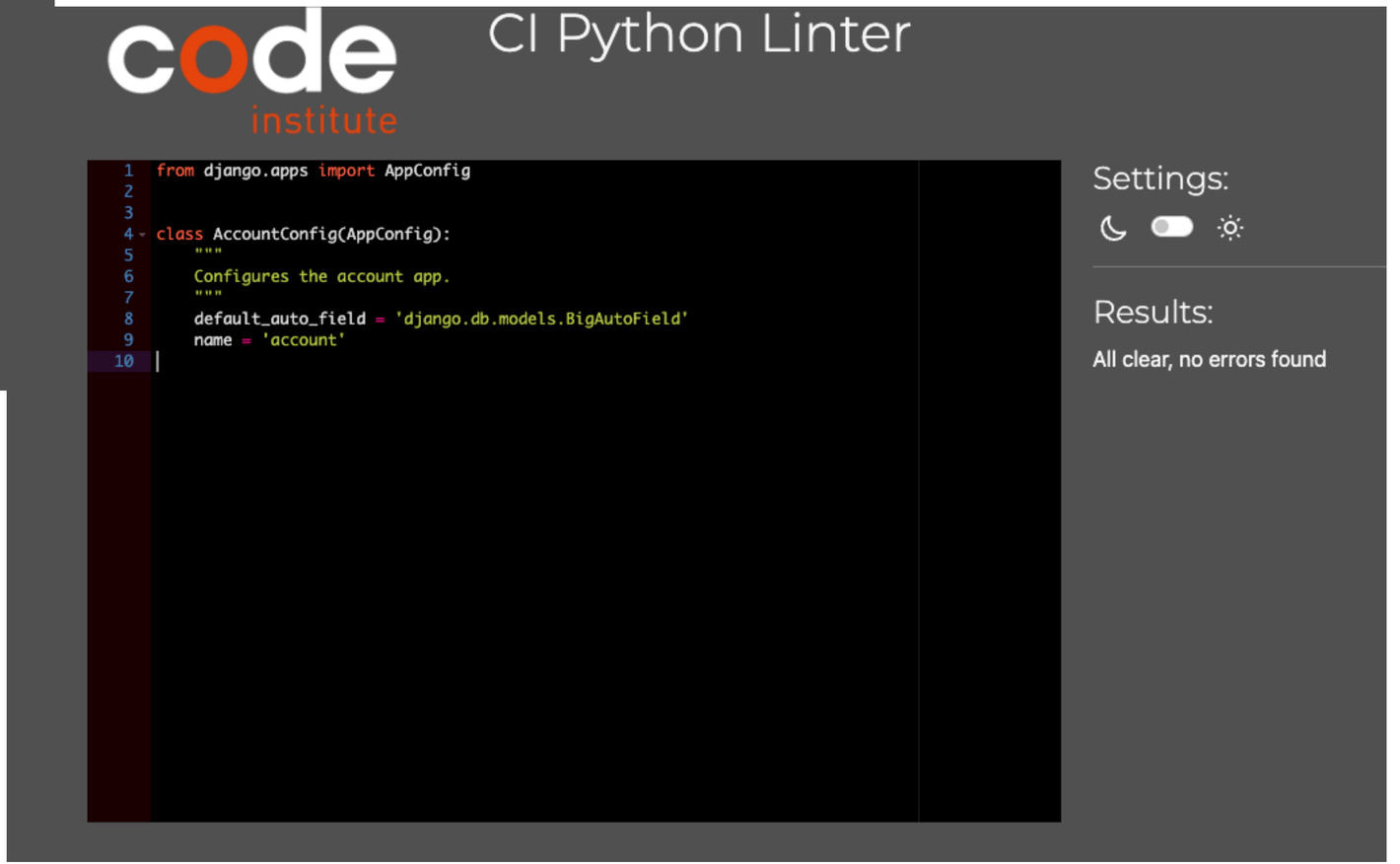
Account

admin.py

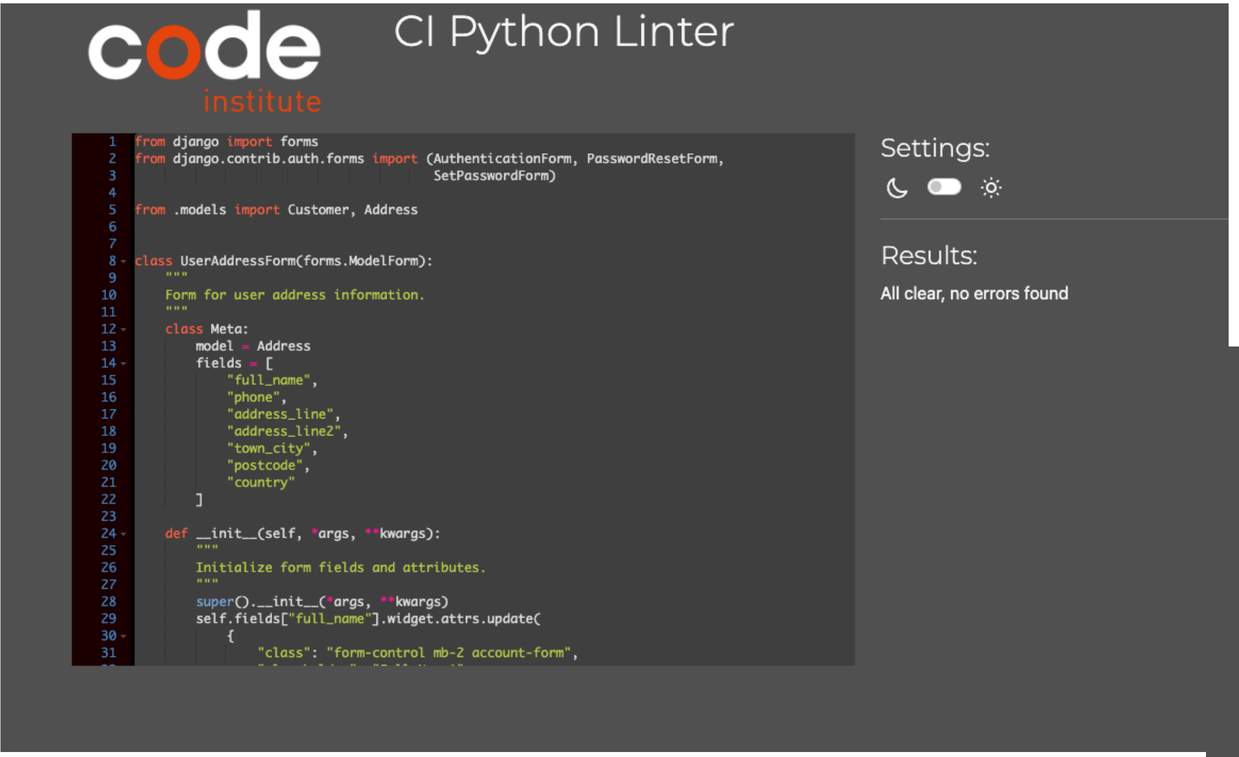


apps.py



Account

forms.py



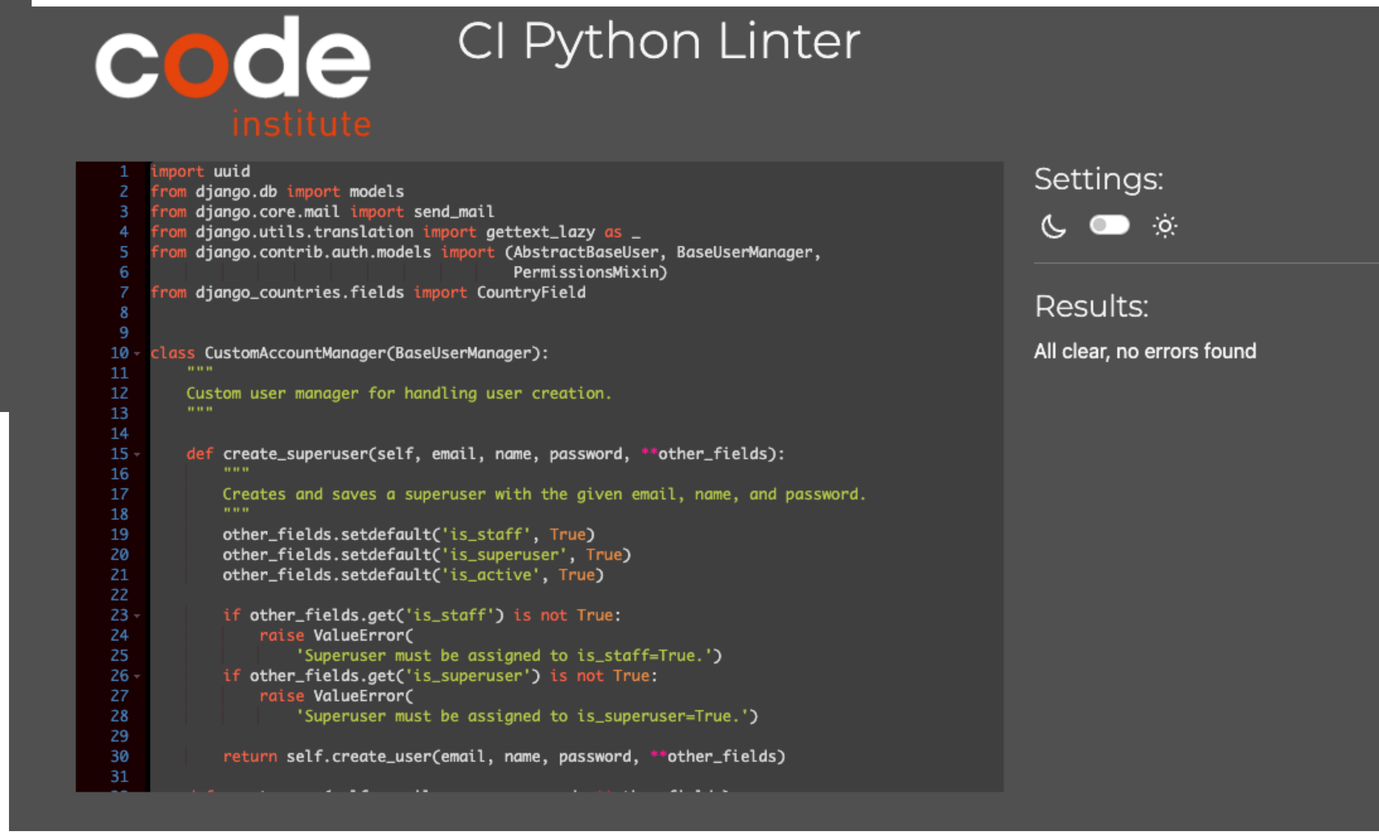

```python
from django import forms
from django.contrib.auth.forms import (AuthenticationForm, PasswordResetForm,
                                       SetPasswordForm)

from .models import Customer, Address


class UserAddressForm(forms.ModelForm):
    """
    Form for user address information.
    """
    class Meta:
        model = Address
        fields = [
            "full_name",
            "phone",
            "address_line",
            "address_line2",
            "town_city",
            "postcode",
            "country"
        ]

    def __init__(self, *args, **kwargs):
        """
        Initialize form fields and attributes.
        """
        super().__init__(*args, **kwargs)
        self.fields["full_name"].widget.attrs.update(
            {
                "class": "form-control mb-2 account-form",
```

models.py




```python
import uuid
from django.db import models
from django.core.mail import send_mail
from django.utils.translation import gettext_lazy as _
from django.contrib.auth.models import (AbstractBaseUser, BaseUserManager,
                                        PermissionsMixin)
from django_countries.fields import CountryField


class CustomAccountManager(BaseUserManager):
    """
    Custom user manager for handling user creation.
    """

    def create_superuser(self, email, name, password, **other_fields):
        """
        Creates and saves a superuser with the given email, name, and password.
        """
        other_fields.setdefault('is_staff', True)
        other_fields.setdefault('is_superuser', True)
        other_fields.setdefault('is_active', True)

        if other_fields.get('is_staff') is not True:
            raise ValueError(
                'Superuser must be assigned to is_staff=True.')
        if other_fields.get('is_superuser') is not True:
            raise ValueError(
                'Superuser must be assigned to is_superuser=True.')

        return self.create_user(email, name, password, **other_fields)
```

Account

tokens.py

CI Python Linter

```python
from django.contrib.auth.tokens import PasswordResetTokenGenerator
from six import text_type


class AccountActivationTokenGenerator(PasswordResetTokenGenerator):
    """
    Token generator for account activation.
    """
    def _make_hash_value(self, user, timestamp):
        """
        Generates a hash value for the token.
        """
        return (
            text_type(user.pk) + text_type(timestamp) +
            text_type(user.is_active)
        )

account_activation_token = AccountActivationTokenGenerator()
```

Settings:

Results:

All clear, no errors found

urls.py

CI Python Linter

```python
"""
URL configuration for account app.

"""

from django.contrib.auth import views as auth_views
from django.urls import path
from django.views.generic import TemplateView

from . import views
from .views import custom_logout
from .forms import (PwdResetConfirmForm, PwdResetForm, UserLoginForm)


app_name = 'account'

urlpatterns = [
    path('login/', auth_views.LoginView.as_view(
        template_name='account/registration/login.html',
        form_class=UserLoginForm), name='login'),
    path('logout/', custom_logout, name='logout'),
    path('register/', views.account_register, name='register'),
    path(
        'activate/<slug:uidb64>/<slug:token>/',
        views.account_activate,
        name='activate'
    ),
    # Reset password
    path('password_reset/', auth_views.PasswordResetView.as_view(
        template_name="account/user/password_reset_form.html",
        success_url='password_reset_email_confirm',
```

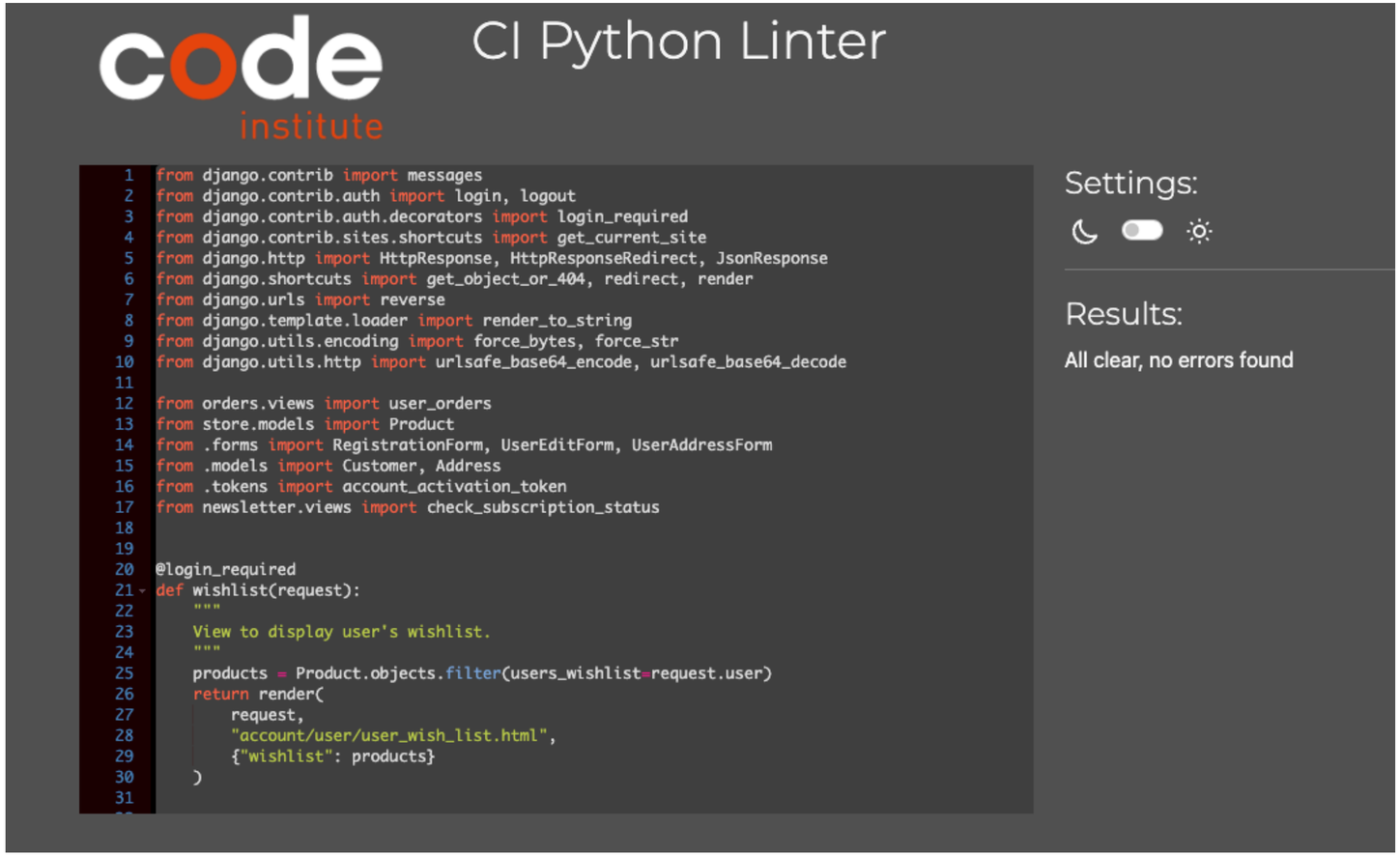
Settings:

Results:

All clear, no errors found

Account

views.py




```python
from django.contrib import messages
from django.contrib.auth import login, logout
from django.contrib.auth.decorators import login_required
from django.contrib.sites.shortcuts import get_current_site
from django.http import HttpResponse, HttpResponseRedirect, JsonResponse
from django.shortcuts import get_object_or_404, redirect, render
from django.urls import reverse
from django.template.loader import render_to_string
from django.utils.encoding import force_bytes, force_str
from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode

from orders.views import user_orders
from store.models import Product
from .forms import RegistrationForm, UserEditForm, UserAddressForm
from .models import Customer, Address
from .tokens import account_activation_token
from newsletter.views import check_subscription_status


@login_required
def wishlist(request):
    """
    View to display user's wishlist.
    """
    products = Product.objects.filter(users_wishlist=request.user)
    return render(
        request,
        "account/user/user_wish_list.html",
        {"wishlist": products}
    )
```