Settings.py


code institute
CI Python Linter
import os
import dj_database_url
from pathlib import Path
if os.path.isfile("env.py"):
import env
BASE_DIR = Path(__file__).resolve().parent.parent
# Security settings
DEBUG = True
ALLOWED_HOSTS = [
"localhost",
"127.0.0.1",
"artera-d6829bf39792.herokuapp.com",
]
CSRF_TRUSTED_ORIGINS = [
"https://artera-d6829bf39792.herokuapp.com",
"https://localhost",
"https://127.0.0.1",
]
SECRET_KEY = os.getenv("SECRET_KEY", "")
# Application definition
INSTALLED_APPS = [
"django.contrib.admin",
"django.contrib.auth",
"django.contrib.contenttypes",
"django.contrib.sessions",
"django.contrib.messages",
"django.contrib.staticfiles",
"django.contrib.sites",
Settings:
Results:
All clear, no errors found

Bag. Context.py


```python
from decimal import Decimal
from django.conf import settings
from django.shortcuts import get_object_or_404
from shop.models import Artwork


def bag_contents(request):
    """Works with the simple bag structure from your views"""

    bag_items = []
    total = Decimal("0")
    product_count = 0
    bag = request.session.get("bag", {})

    for item_id, item_data in bag.items():
        artwork = get_object_or_404(Artwork, pk=item_id)
        unit_price = Decimal(getattr(artwork, "price", 0) or 0)

        # If item has sizes
        if (
            isinstance(item_data, dict)
            and "sizes" in item_data
            and isinstance(item_data["sizes"], dict)
        ):
            for size, quantity in item_data["sizes"].items():
                qty = int(quantity) if quantity else 1
                line_total = unit_price * qty
                total += line_total
                product_count += qty
                bag_items.append(
                    {
                        "item_id": item_id,
                        "quantity": qty,
                        "artwork": artwork,
```

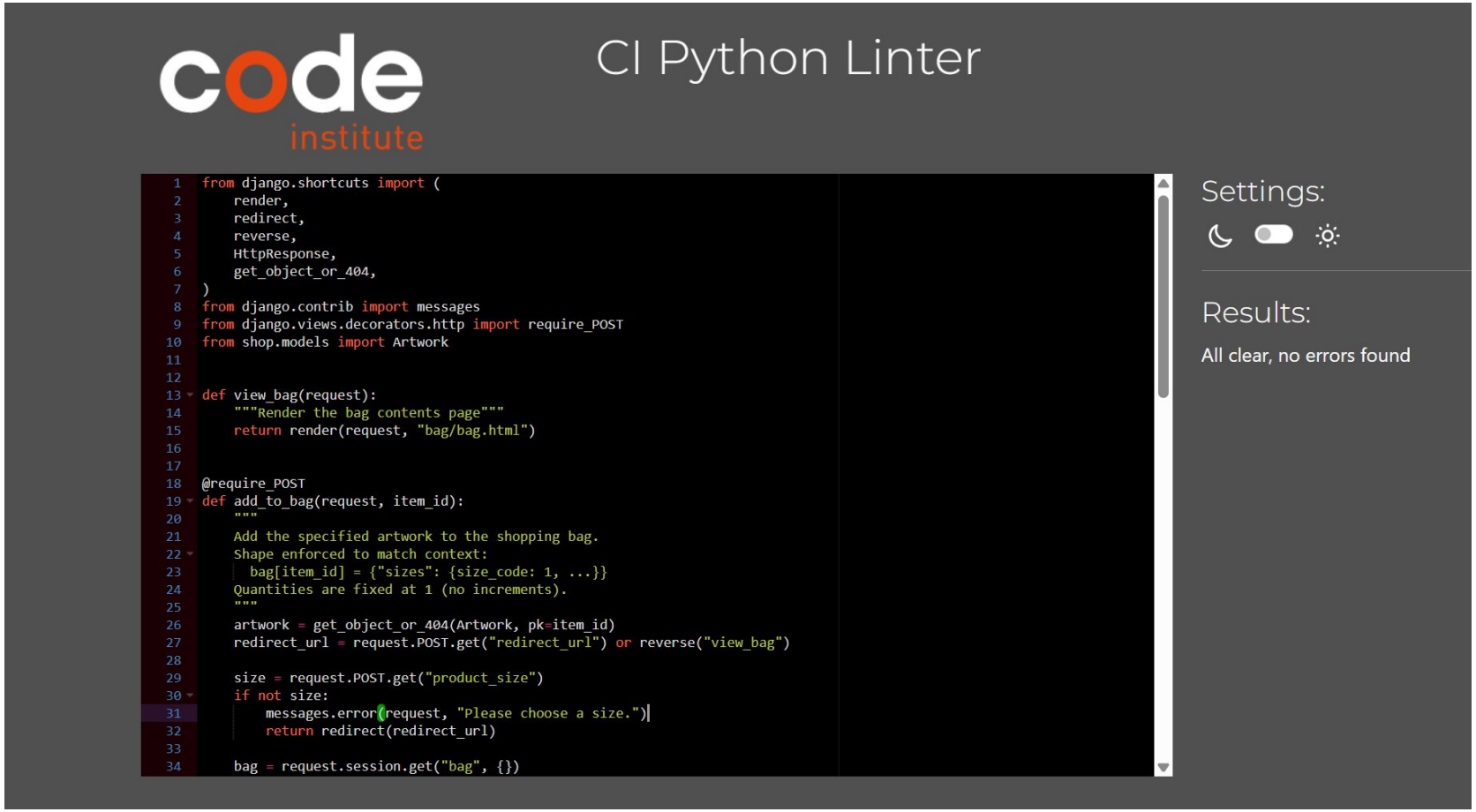
Bag. Apps.py


code institute
CI Python Linter
1 from django.apps import AppConfig
2
3
4 class BagConfig(AppConfig):
5 default_auto_field = 'django.db.models.BigAutoField'
6 name = 'bag'
7
Settings:
Results:
All clear, no errors found

Bag. Urls.py


code institute
CI Python Linter
from django.urls import path
from . import views
urlpatterns = [
path('', views.view_bag, name='view_bag'),
path('add/<item_id>/', views.add_to_bag, name='add_to_bag'),
path(
"remove/<int:item_id>/", views.remove_from_bag, name="remove_from_bag",
),
]
Settings:
Results:
All clear, no errors found

Bag. Views.py



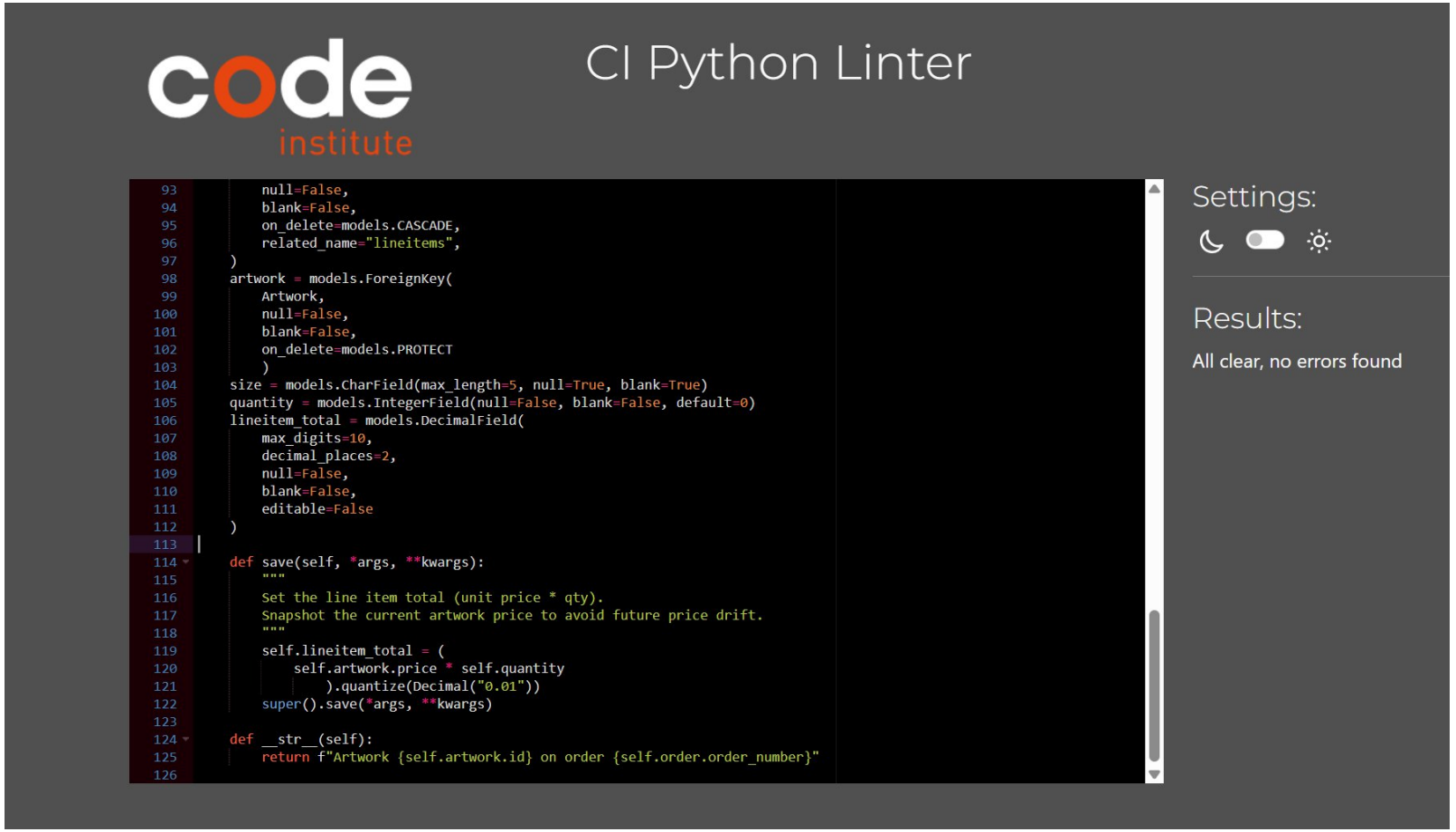

```python
from django.shortcuts import (
    render,
    redirect,
    reverse,
    HttpResponse,
    get_object_or_404,
)
from django.contrib import messages
from django.views.decorators.http import require_POST
from shop.models import Artwork


def view_bag(request):
    """Render the bag contents page"""
    return render(request, "bag/bag.html")


@require_POST
def add_to_bag(request, item_id):
    """
    Add the specified artwork to the shopping bag.
    Shape enforced to match context:
      bag[item_id] = {"sizes": {size_code: 1, ...}}
    Quantities are fixed at 1 (no increments).
    """
    artwork = get_object_or_404(Artwork, pk=item_id)
    redirect_url = request.POST.get("redirect_url") or reverse("view_bag")

    size = request.POST.get("product_size")
    if not size:
        messages.error(request, "Please choose a size.")
        return redirect(redirect_url)

    bag = request.session.get("bag", {})
```

Checkout. Admin.py

Checkout. Apps.py

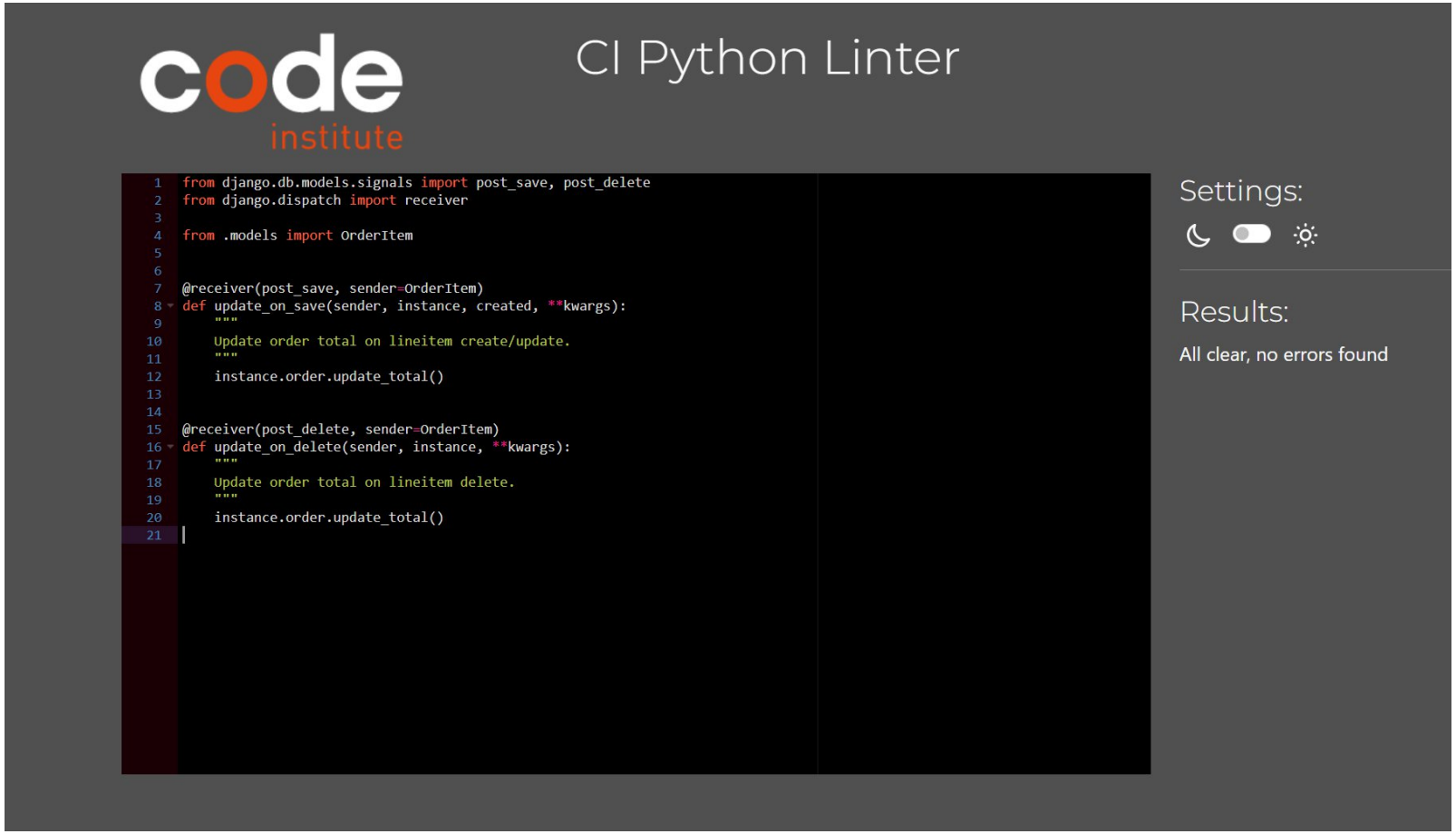
Checkout. Forms.py

Checkout. Models.py



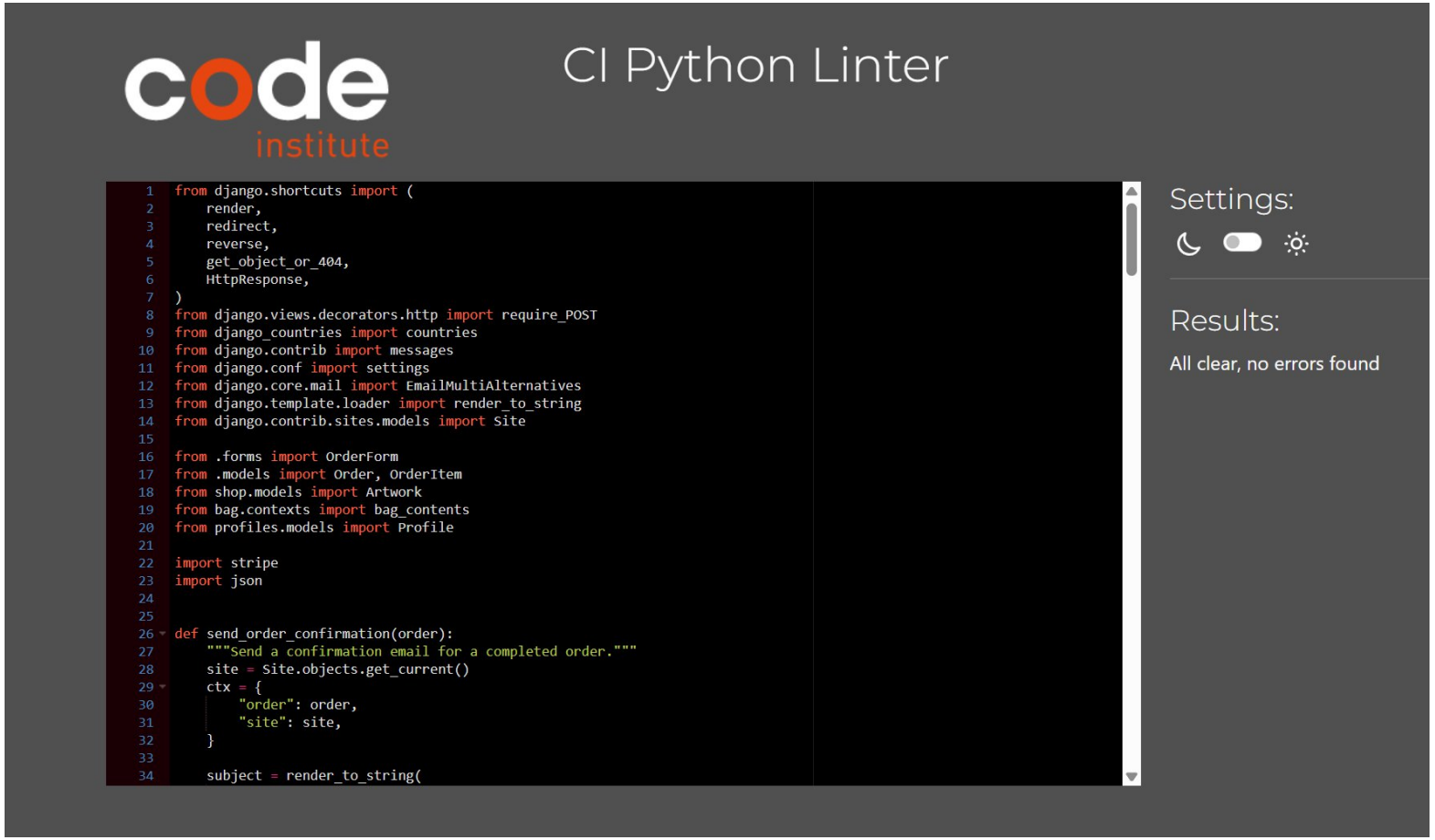

```
        null=False,
        blank=False,
        on_delete=models.CASCADE,
        related_name="lineitems",
    )
    artwork = models.ForeignKey(
        Artwork,
        null=False,
        blank=False,
        on_delete=models.PROTECT
        )
    size = models.CharField(max_length=5, null=True, blank=True)
    quantity = models.IntegerField(null=False, blank=False, default=0)
    lineitem_total = models.DecimalField(
        max_digits=10,
        decimal_places=2,
        null=False,
        blank=False,
        editable=False
    )

    def save(self, *args, **kwargs):
        """
        Set the line item total (unit price * qty).
        Snapshot the current artwork price to avoid future price drift.
        """
        self.lineitem_total = (
            self.artwork.price * self.quantity
                ).quantize(Decimal("0.01"))
        super().save(*args, **kwargs)

    def __str__(self):
        return f"Artwork {self.artwork.id} on order {self.order.order_number}"
```

Checkout. Signals.py



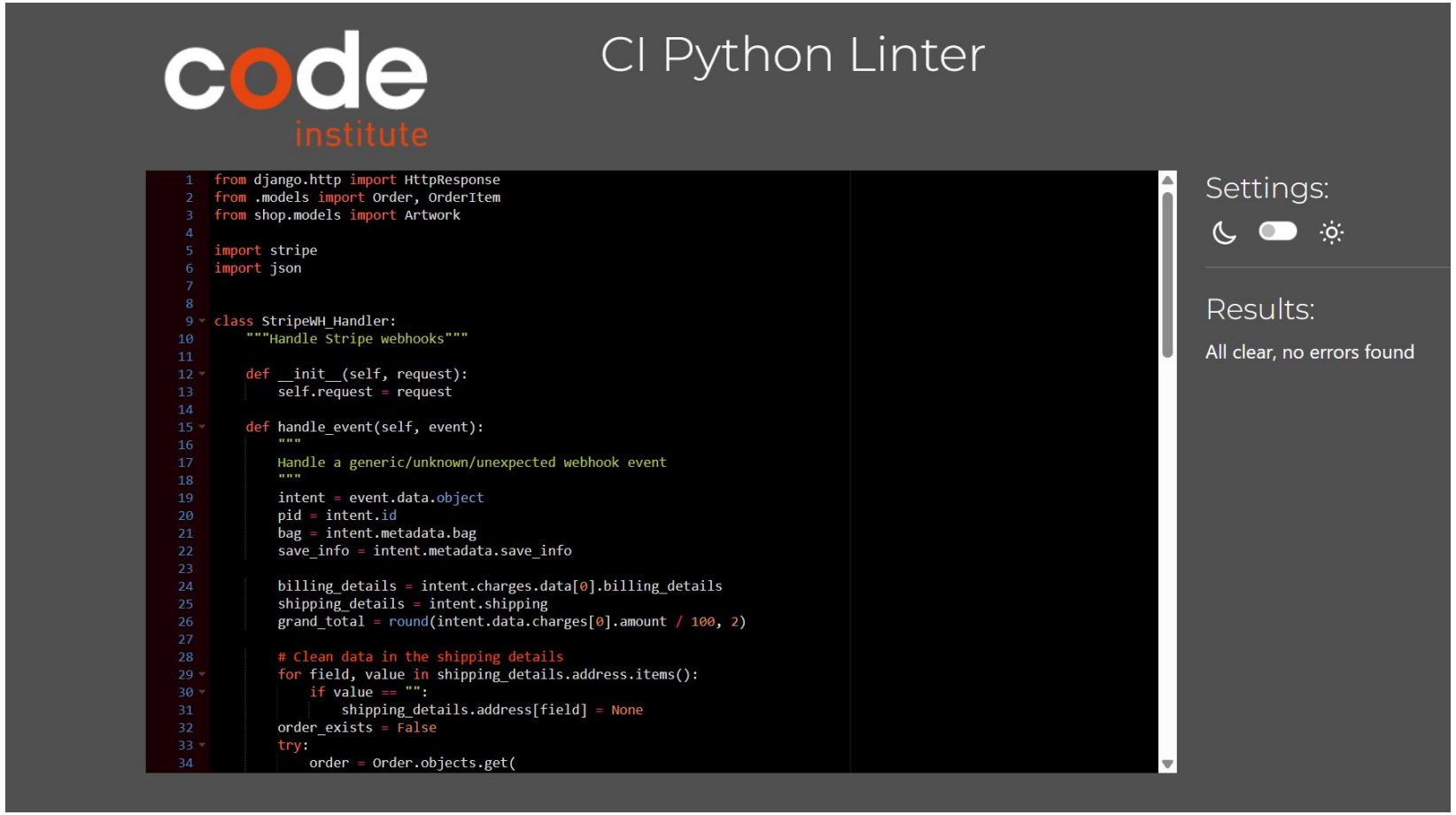
Checkout. Urls.py

Checkout. Views.py




```python
from django.shortcuts import (
    render,
    redirect,
    reverse,
    get_object_or_404,
    HttpResponse,
)
from django.views.decorators.http import require_POST
from django_countries import countries
from django.contrib import messages
from django.conf import settings
from django.core.mail import EmailMultiAlternatives
from django.template.loader import render_to_string
from django.contrib.sites.models import Site

from .forms import OrderForm
from .models import Order, OrderItem
from shop.models import Artwork
from bag.contexts import bag_contents
from profiles.models import Profile

import stripe
import json


def send_order_confirmation(order):
    """Send a confirmation email for a completed order."""
    site = Site.objects.get_current()
    ctx = {
        "order": order,
        "site": site,
    }

    subject = render_to_string(
```

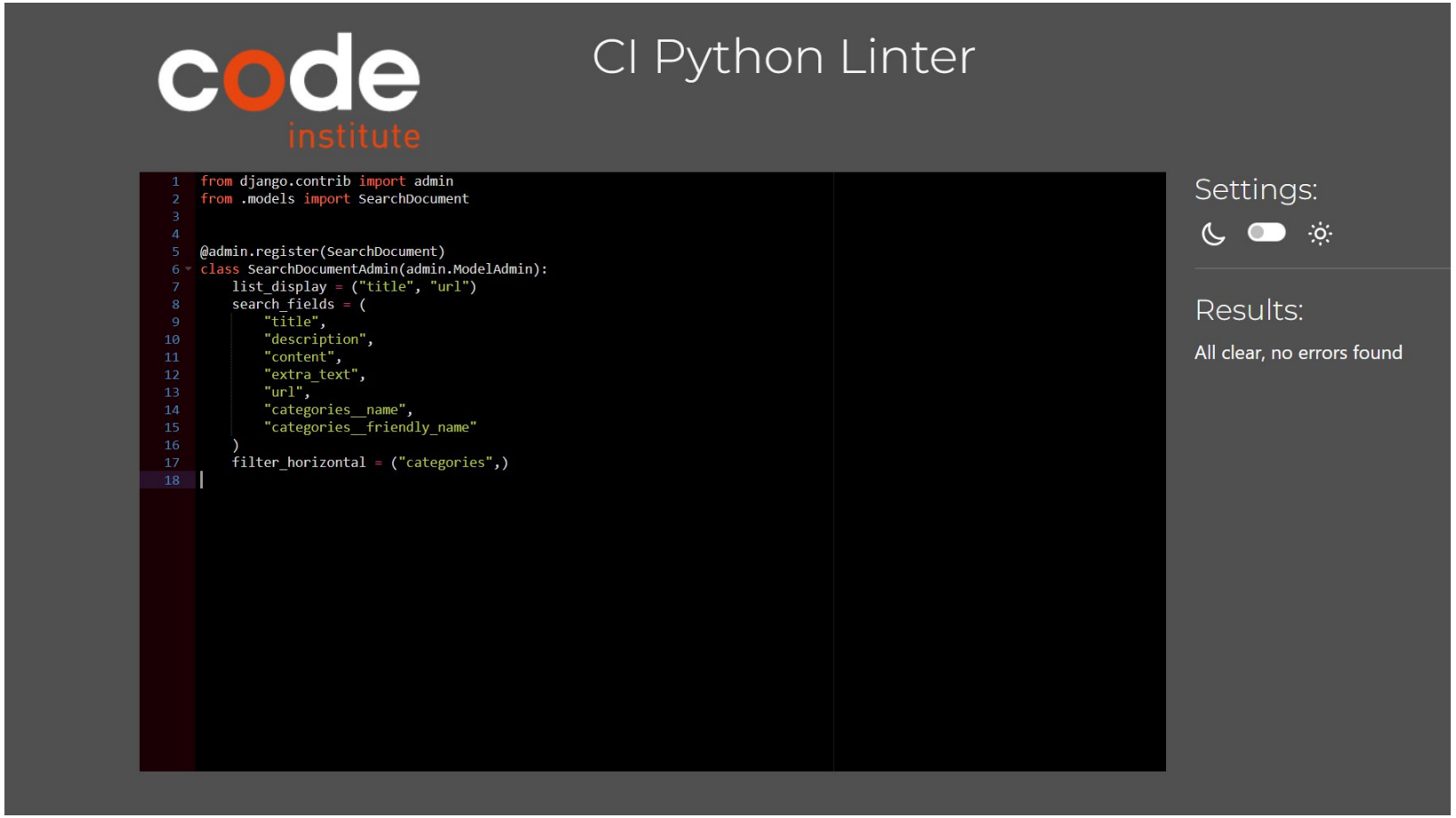
Checkout. Webhook_handler.py




```python
from django.http import HttpResponse
from .models import Order, OrderItem
from shop.models import Artwork

import stripe
import json


class StripeWH_Handler:
    """Handle Stripe webhooks"""

    def __init__(self, request):
        self.request = request

    def handle_event(self, event):
        """
        Handle a generic/unknown/unexpected webhook event
        """
        intent = event.data.object
        pid = intent.id
        bag = intent.metadata.bag
        save_info = intent.metadata.save_info

        billing_details = intent.charges.data[0].billing_details
        shipping_details = intent.shipping
        grand_total = round(intent.data.charges[0].amount / 100, 2)

        # Clean data in the shipping details
        for field, value in shipping_details.address.items():
            if value == "":
                shipping_details.address[field] = None
        order_exists = False
        try:
            order = Order.objects.get(
```

Checkout. Webhooks.py

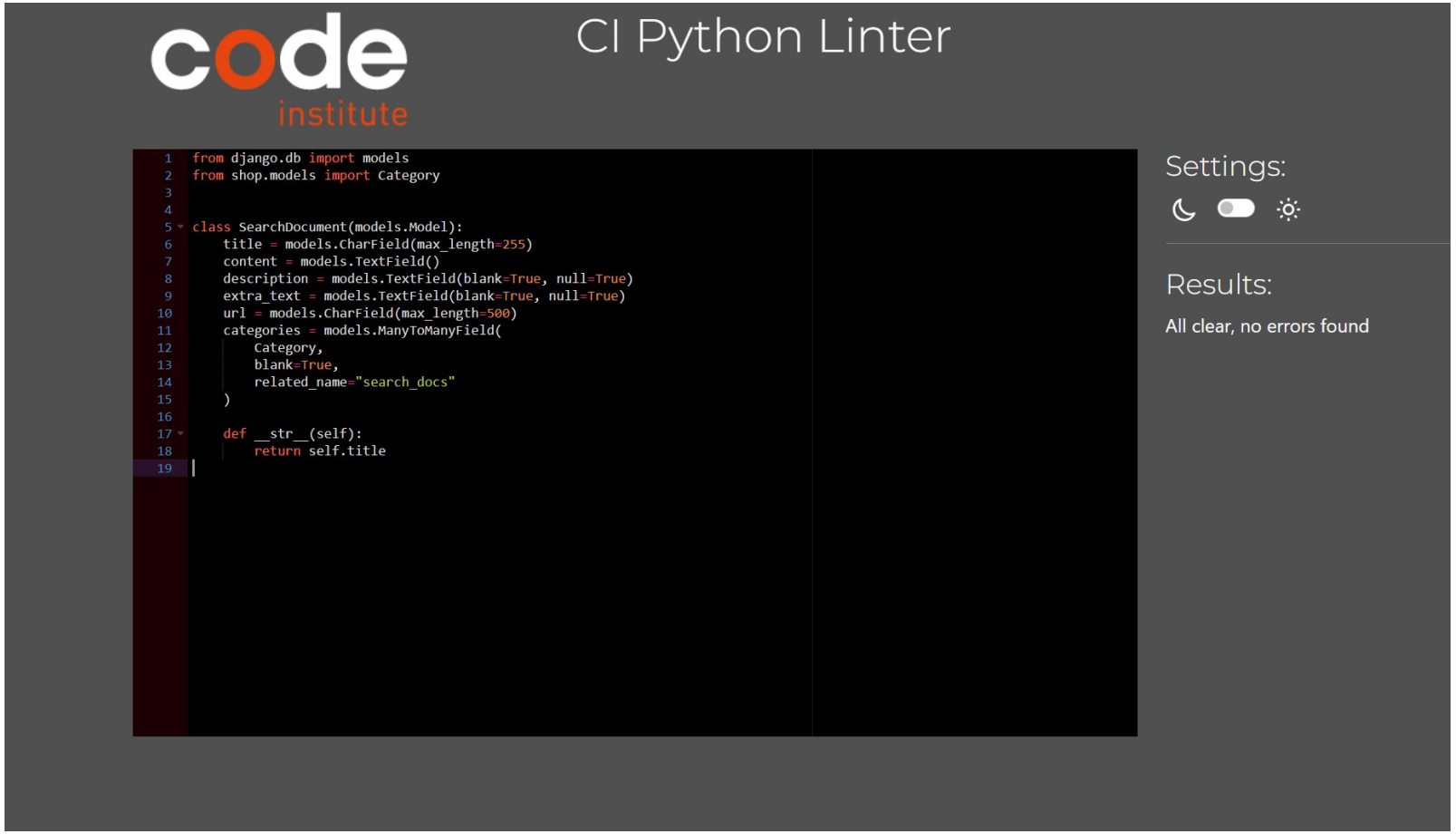
Home. Admin.py



Home. Apps.py


code institute
CI Python Linter
1 from django.apps import AppConfig
2
3
4 class HomeConfig(AppConfig):
5 default_auto_field = 'django.db.models.BigAutoField'
6 name = 'home'
7
Settings:
Results:
All clear, no errors found

Home. Models.py



Home. Urls.py


code institute
CI Python Linter
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5 path('', views.homepege, name='home'),
6 path('artwork-search/', views.artwork_search, name='artwork_search'),
7]
8
Settings:
Results:
All clear, no errors found

Home. Views.py

Profiles. Apps.py

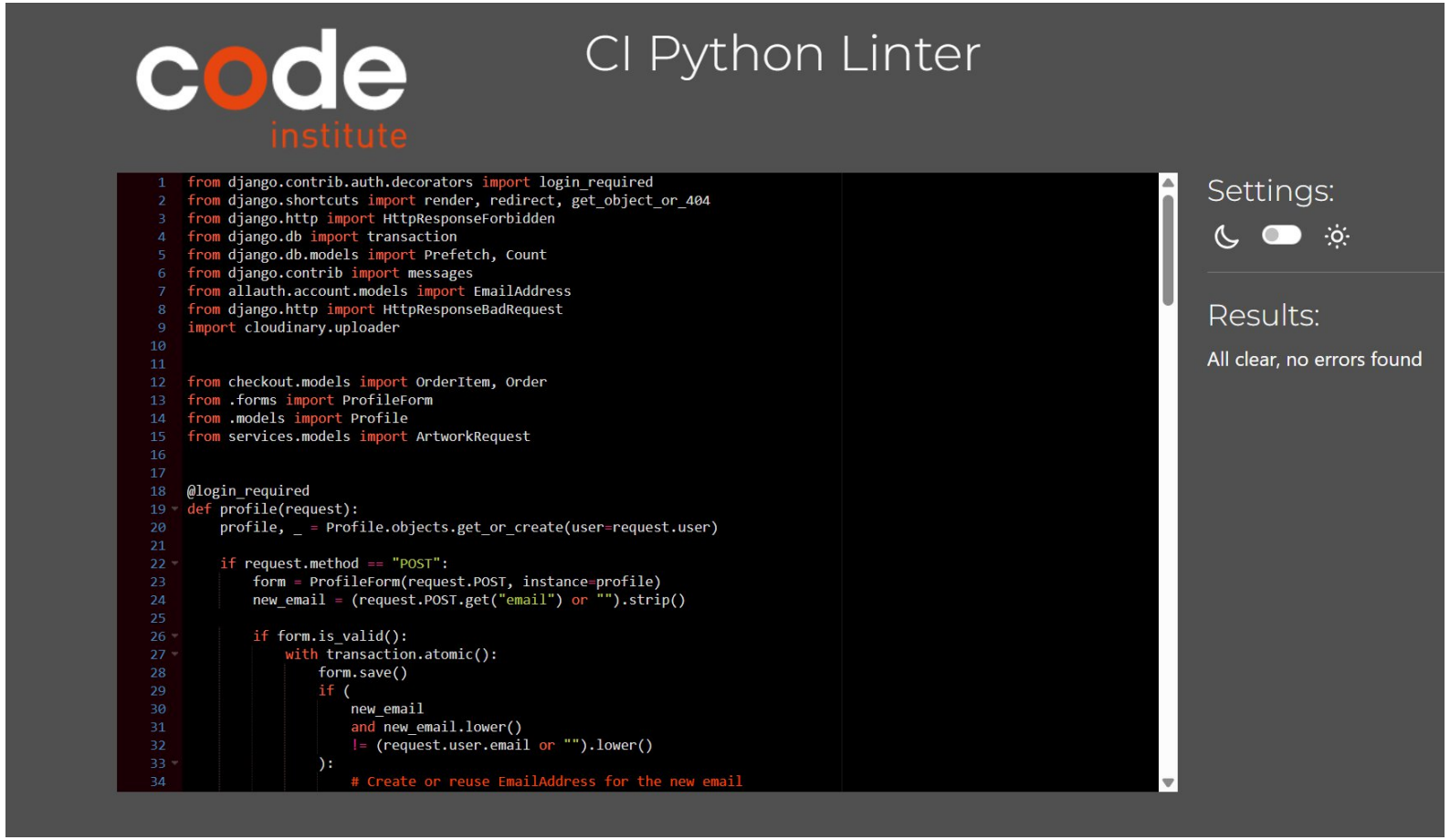
Profiles. Forms.py

Profiles. Models.py

Profiles. Signals.py

Profiles. Urls.py

Profiles. Views.py

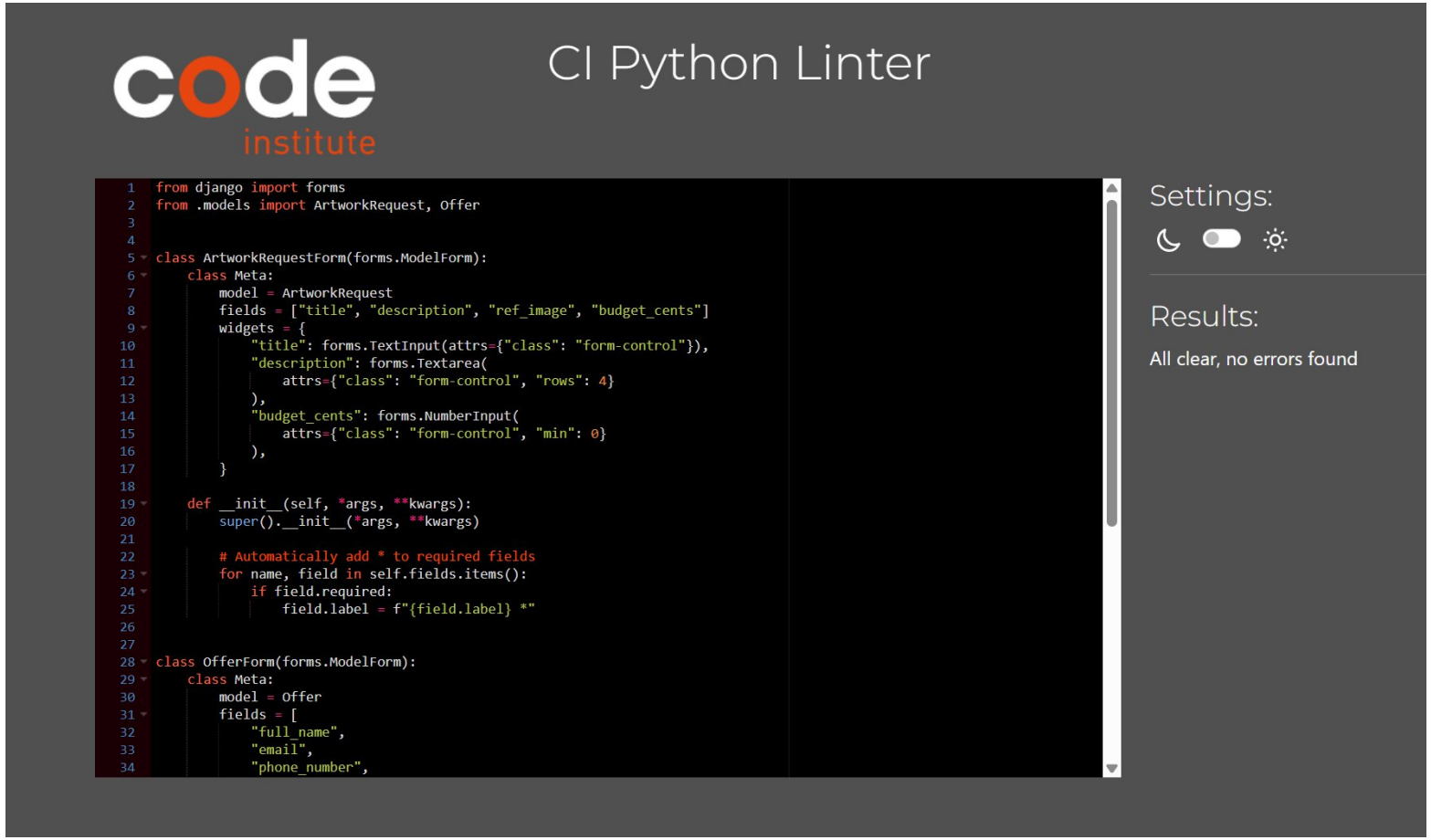



```python
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, redirect, get_object_or_404
from django.http import HttpResponseForbidden
from django.db import transaction
from django.db.models import Prefetch, Count
from django.contrib import messages
from allauth.account.models import EmailAddress
from django.http import HttpResponseBadRequest
import cloudinary.uploader


from checkout.models import OrderItem, Order
from .forms import ProfileForm
from .models import Profile
from services.models import ArtworkRequest


@login_required
def profile(request):
    profile, _ = Profile.objects.get_or_create(user=request.user)

    if request.method == "POST":
        form = ProfileForm(request.POST, instance=profile)
        new_email = (request.POST.get("email") or "").strip()

        if form.is_valid():
            with transaction.atomic():
                form.save()
                if (
                    new_email
                    and new_email.lower()
                    != (request.user.email or "").lower()
                ):
                    # Create or reuse EmailAddress for the new email
```

Services. Admin.py

Services. Apps.py

Services. Forms.py



Services. Models.py

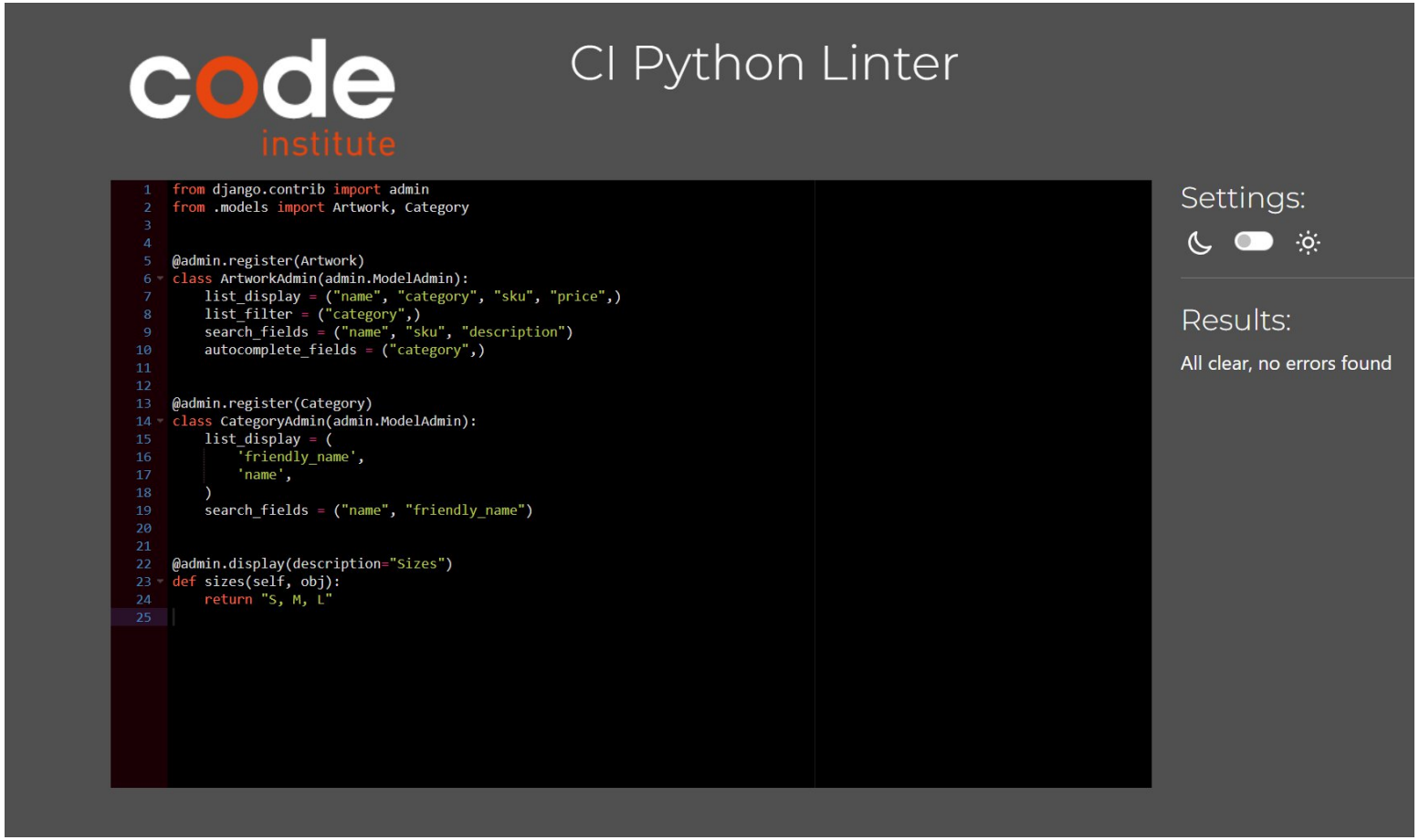

```python
from django.db import models
from django.conf import settings
from cloudinary.models import CloudinaryField


class ArtworkRequest(models.Model):
    STATUS = [
        ("in_review", "In review"),
        ("accepted", "Accepted"),
        ("rejected", "Rejected"),
    ]
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
        related_name="service_requests"
    )
    title = models.CharField(max_length=120, blank=False)
    description = models.TextField(blank=False, max_length=1000)
    ref_image = CloudinaryField('image', folder='requests', blank=True)
    budget_cents = models.DecimalField(
        max_digits=10,
        decimal_places=2,
        null=True,
        blank=True
    )
    status = models.CharField(
        max_length=20,
        choices=STATUS,
        default="in_review"
    )
    created_at = models.DateTimeField(auto_now_add=True)


class Offer(models.Model):
```

Services. Urls.py

Services. Views.py


```python
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from .forms import ArtworkRequestForm, OfferForm
from .models import ArtworkRequest
from .models import Offer


@login_required
def artwork_request(request):
    if request.method == "POST":
        form = ArtworkRequestForm(request.POST, request.FILES)
        if form.is_valid():
            obj = form.save(commit=False)
            obj.user = request.user
            obj.save()
            messages.success(request, "Request submitted for review.")
            return redirect("request_detail", pk=obj.pk)
        messages.error(request, "Please fix the errors below.")
    else:
        form = ArtworkRequestForm()

    return render(request, "services/request_form.html", {"form": form})


@login_required
def request_detail(request, pk):
    obj = get_object_or_404(ArtworkRequest, pk=pk, user=request.user)
    return render(
        request,
        "services/request_detail.html", {"request_obj": obj}
    )
```

Shop. Admin.py

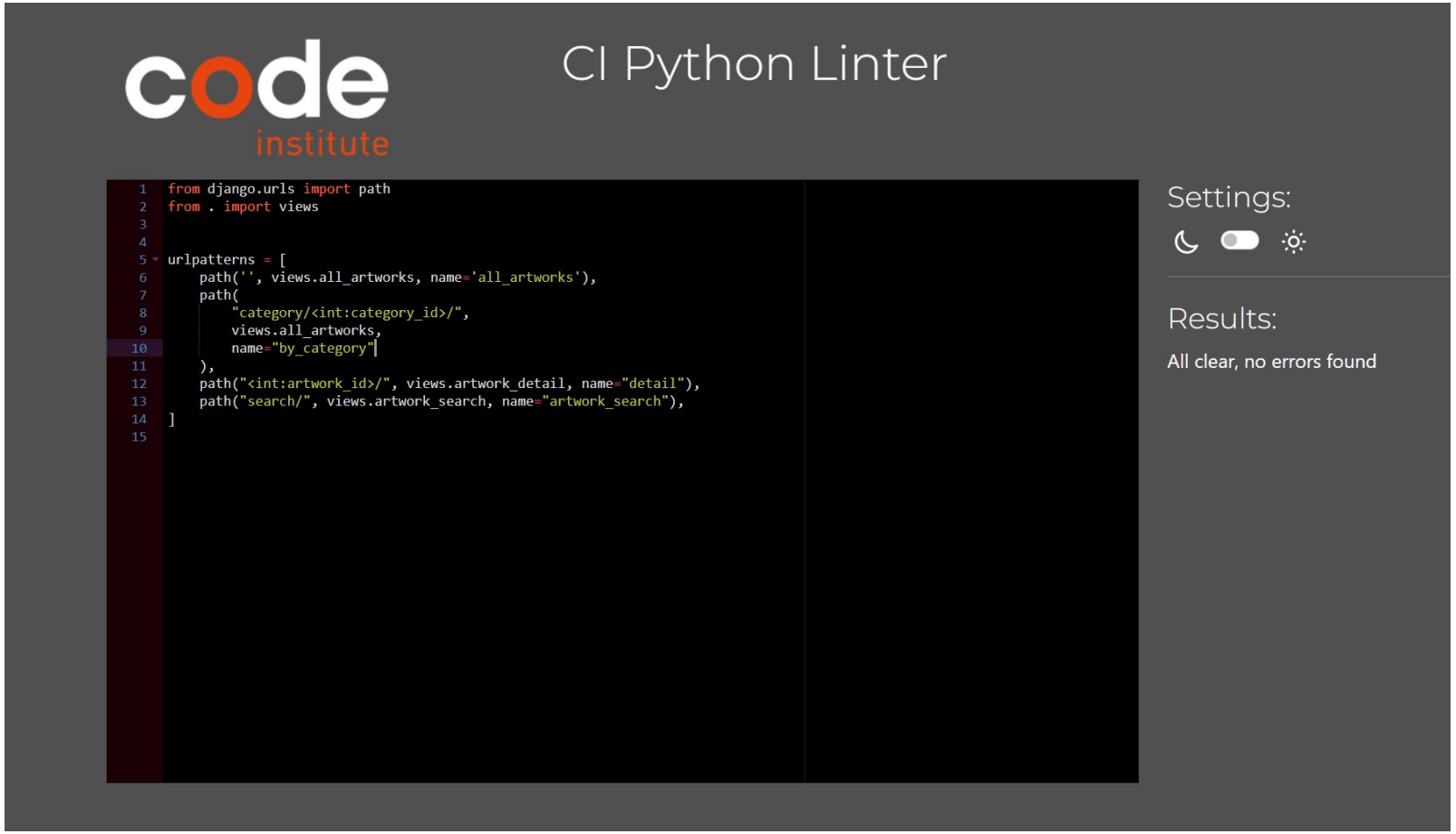


Shop. Apps.py


code institute
CI Python Linter
1 from django.apps import AppConfig
2
3
4 class ShopConfig(AppConfig):
5 default_auto_field = 'django.db.models.BigAutoField'
6 name = 'shop'
7
Settings:
Results:
All clear, no errors found

Shop. Models.py

Shop. Urls.py


code institute
CI Python Linter
from django.urls import path
from . import views
urlpatterns = [
path('', views.all_artworks, name='all_artworks'),
path(
"category/<int:category_id>/",
views.all_artworks,
name="by_category"
),
path("<int:artwork_id>/", views.artwork_detail, name="detail"),
path("search/", views.artwork_search, name="artwork_search"),
]
Settings:
Results:
All clear, no errors found

Shop. Views.py


```python
from django.shortcuts import render, redirect, reverse, get_object_or_404
from django.contrib import messages
from django.db.models import Q
from django.db.models.functions import Lower
from decimal import Decimal
from django.utils import timezone
from datetime import timedelta
from django.db.models.functions import Random

from .models import Artwork, Category, SIZE_CHOICES, SIZE_SURCHARGE


def all_artworks(request, category_id=None):
    artworks = Artwork.objects.select_related("category").order_by(Random())
    categories = Category.objects.order_by("name")

    if category_id:
        artworks = artworks.filter(category_id=category_id)
    query = None
    current_categories = None

    if "category" in request.GET:
        names = [
            c.strip()
            for c in request.GET["category"].split(",") if c.strip()]
        artworks = artworks.filter(category__name__in=names)
        current_categories = Category.objects.filter(name__in=names)

    context = {
        "artworks": artworks,
        "search_term": query,
        "current_categories": current_categories,
        "SIZE_SURCHARGE": SIZE_SURCHARGE,
        "categories": categories,
```