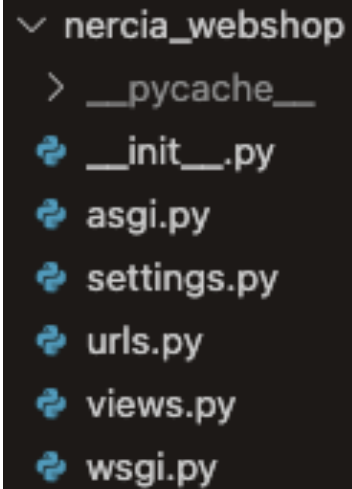


Python Validation

Project: nercia_webshop



A screenshot of a file explorer interface with a dark background. It shows a tree view of a project named 'nercia_webshop'. Under this project, there is a sub-entry '> __pycache__' and a list of Python files, each preceded by a blue icon representing a file. The files are: '__init__.py', 'asgi.py', 'settings.py', 'urls.py', 'views.py', and 'wsgi.py'.

- ▼ nercia_webshop
 - > __pycache__
 - 🔗 __init__.py
 - 🔗 asgi.py
 - 🔗 settings.py
 - 🔗 urls.py
 - 🔗 views.py
 - 🔗 wsgi.py

Python Validation

Nercia_webshop/views.py

```
1  """Views to handle errors"""
2  from django.shortcuts import render
3
4
5  def handler404(request, exception):
6      """ Error Handler 404 - Page Not Found """
7      return render(request, "errors/404.html", status=404)
8
9
10 def handler500(request):
11     """ Error Handler 500 - Internal Server Error """
12     return render(request, "errors/500.html", status=500)
13 |
```

Settings:



Results:

All clear, no errors found

Python Validation

Nercia_webshop/settings.py

```
1 """
2 Django settings for nercia_webshop project.
3
4 Generated by 'django-admin startproject' using Django 3.2.23.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/3.2/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/3.2/ref/settings/
11 """
12
13 from pathlib import Path
14 from decouple import config
15
16
17 import os
18 import dj_database_url
19
20 if os.path.exists('env.py'):
21     import env
22
23 # Build paths inside the project like this: BASE_DIR / 'subdir'.
24 BASE_DIR = Path(__file__).resolve().parent.parent
25
26
27 # Quick-start development settings - unsuitable for production
28 # See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
29
30 # SECURITY WARNING: keep the secret key used in production secret!
31 SECRET_KEY = os.environ.get("SECRET_KEY")
```

Settings:



Results:

All clear, no errors found

Python Validation

Nercia_webshop/urls.py

```
1 from django.contrib import admin
2 from django.urls import path, include
3 from django.conf import settings
4 from django.conf.urls.static import static
5
6 from .views import handler404, handler500
7
8 urlpatterns = [
9     path('admin/', admin.site.urls),
10    path('accounts/', include('allauth.urls')),
11    path('', include('home.urls')),
12    path('products/', include('products.urls')),
13    path('bag/', include('bag.urls')),
14    path('profile/', include('profiles.urls')),
15    path('checkout/', include('checkout.urls')),
16    path('newsletters/', include('newsletters.urls')),
17    path('contact/', include('contact.urls')),
18 ] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
19
20 handler404 = 'nercia_webshop.views.handler404'
21 handler500 = 'nercia_webshop.views.handler500'
22
```

Settings:

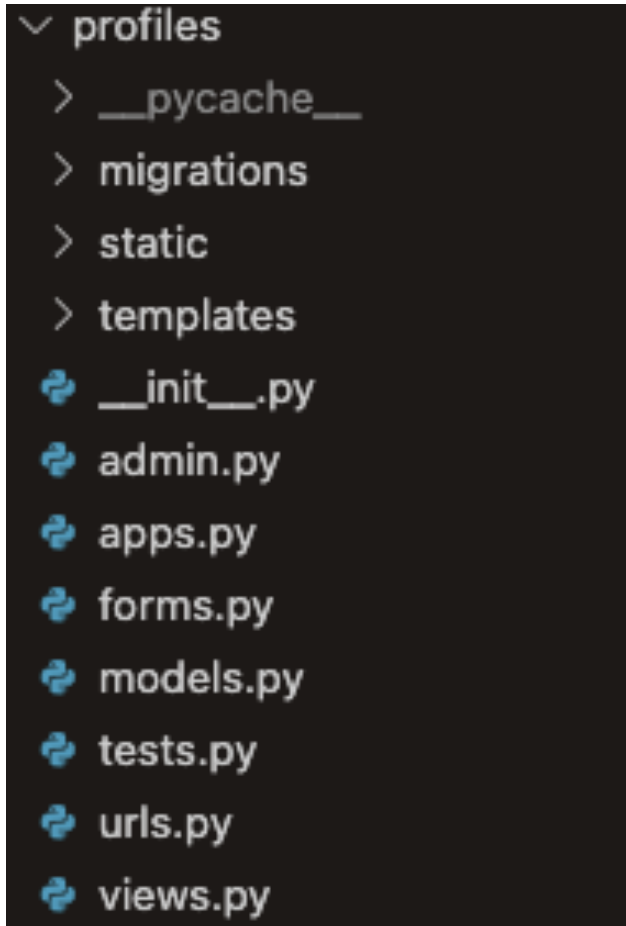


Results:

All clear, no errors found

Python Validation

App: profiles



Python Validation

profiles/admin.py

```
5
6 class UserProfileInline(admin.StackedInline):
7     """ Sets up admin panel for user emails """
8     model = UserProfile
9
10
11 class CustomUserAdmin(UserAdmin):
12     """ Sets up admin panel for user profile info """
13     model = CustomUser
14     list_display = (
15         'email', 'first_name', 'last_name', 'is_staff', 'is_active',
16     )
17     list_filter = ('email', 'is_staff', 'is_active',)
18     fieldsets = (
19         (None, {'fields': ('email', 'password')}),
20         ('Personal Info', {'fields': ('first_name', 'last_name')}),
21         ('Permissions', {'fields': ('is_staff', 'is_active')}),
22     )
23     add_fieldsets = (
24         (None, {
25             'classes': ('wide',),
26             'fields': (
27                 'email', 'password1', 'password2', 'is_staff', 'is_active'
28             ),
29         }),
30     )
31     search_fields = ('email',)
32     ordering = ('email',)
33
34
35 admin.site.register(CustomUser, CustomUserAdmin)
36 admin.site.register(UserProfile)
```

Settings:



Results:

All clear, no errors found

Python Validation

profiles/forms.py

```
7
8 class CustomSignupForm(SignupForm):
9     """ Removes username from signup """
10    def __init__(self, *args, **kwargs):
11        super(CustomSignupForm, self).__init__(*args, **kwargs)
12        del self.fields['username']
13
14
15 class UserProfileForm(forms.ModelForm):
16     """ Sets up user profile form """
17     newsletter_subscription = forms.BooleanField(
18         label='Subscribe to Newsletter',
19         required=False,
20         initial=True,
21         widget=forms.CheckboxInput(attrs={'class': 'stripe-style-input'}),
22     )
23
24    class Meta:
25        model = UserProfile
26        fields = [
27            'first_name',
28            'last_name',
29            'email',
30            'phone_number',
31            'company_name',
32            'org_num',
33            'street_address1',
34            'street_address2',
35            'postcode',
36            'city',
37            'country',
38            ...
39
```

Settings:



Results:

All clear, no errors found

Python Validation

profiles/models.py

```
43 class CustomUser(AbstractUser):
44     """ Custom signup form to remove username and use email as 'username' """
45     email = models.EmailField(_('email address'), unique=True)
46
47     USERNAME_FIELD = 'email'
48     REQUIRED_FIELDS = []
49
50     objects = CustomUserManager()
51
52
53 class UserProfile(models.Model):
54     """ User profile model """
55     user = models.OneToOneField(
56         CustomUser,
57         on_delete=models.CASCADE,
58         unique=True
59     )
60     first_name = models.CharField(max_length=30)
61     last_name = models.CharField(max_length=30)
62     email = models.EmailField()
63     phone_number = models.CharField(max_length=20, null=False, blank=False)
64     company_name = models.CharField(max_length=100, null=False, blank=False)
65     org_num = models.CharField(max_length=20, null=False, blank=False)
66     street_address1 = models.CharField(max_length=80, null=False, blank=False)
67     street_address2 = models.CharField(max_length=80, null=True, blank=True)
68     postcode = models.CharField(max_length=20, null=False, blank=False)
69     city = models.CharField(max_length=40, null=False, blank=False)
70     country = CountryField(blank_label='Country', null=False, blank=False)
71     invoice_email = models.EmailField(max_length=254, null=False, blank=False)
72     newsletter_subscription = models.BooleanField(
73         default=False, null=False, blank=False
```

Settings:



Results:

All clear, no errors found

Python Validation

profiles/urls.py

```
1 from django.urls import path
2 from . import views
3 from allauth.account.views import SignupView
4 from .forms import CustomSignupForm, DeleteAccountForm
5
6 urlpatterns = [
7     path('', views.profile, name='profile'),
8     path(
9         'accounts/signup/',
10        SignupView.as_view(form_class=CustomSignupForm),
11        name='account_signup'
12    ),
13    path(
14        'order_history/<order_number>/',
15        views.order_history,
16        name='order_history'
17    ),
18    path(
19        'delete_profile/<int:user_profile_id>/',
20        views.delete_profile,
21        name='delete_profile'
22    ),
23 ]
24 |
```

Settings:



Results:

All clear, no errors found

Python Validation

profiles/views.py

```
12
13 class CustomSignUpView(SignupView):
14     """ Sets up custom signup view """
15     form_class = CustomSignupForm
16
17
18 @login_required
19 def profile(request):
20     """ A view for profile page """
21     try:
22         user_profile = request.user.userprofile
23     except UserProfile.DoesNotExist:
24         user_profile, created = UserProfile.objects.get_or_create(
25             user=request.user
26         )
27
28     orders = user_profile.orders.all()
29
30     initial_data = {
31         'first_name': user_profile.first_name,
32         'last_name': user_profile.last_name,
33         'email': request.user.email,
34         'phone_number': user_profile.phone_number,
35         'company_name': user_profile.company_name,
36         'org_num': user_profile.org_num,
37         'street_address1': user_profile.street_address1,
38         'street_address2': user_profile.street_address2,
39         'postcode': user_profile.postcode,
40         'city': user_profile.city,
41         'country': user_profile.country,
42         'invoice_email': user_profile.invoice_email,
43     }
```

Settings:

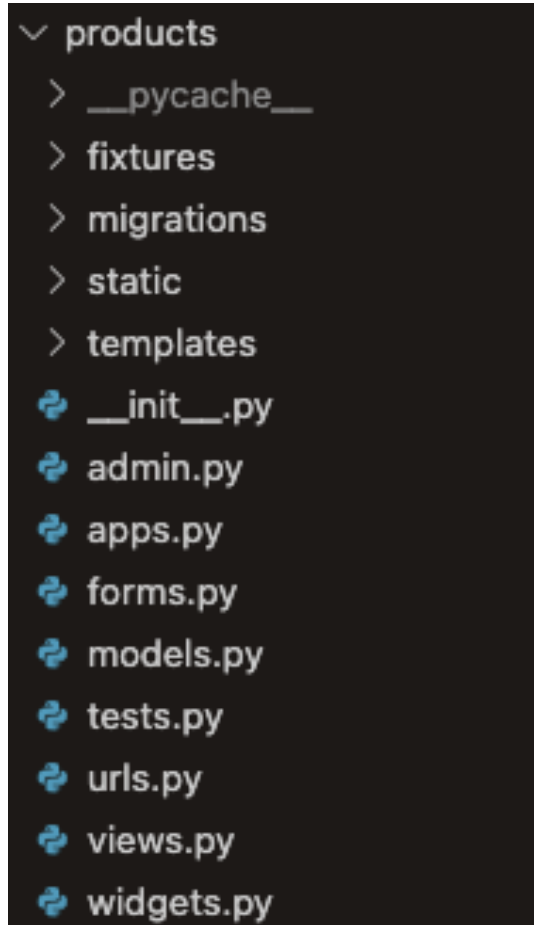


Results:

All clear, no errors found

Python Validation

App: products



Python Validation

products/admin.py

```
5 class CategoryAdmin(admin.ModelAdmin):
6     list_display = (
7         'friendly_name',
8         'name',
9     )
10
11
12 class ProductAdmin(admin.ModelAdmin):
13     list_display = (
14         'name',
15         'duration',
16         'online_onsite',
17         'price',
18     )
19
20     ordering = ('name',)
21
22
23 class ProductContentAdmin(admin.ModelAdmin):
24     list_display = (
25         'product',
26         'title',
27         'day',
28     )
29
30     ordering = ('product',)
31
32
33 admin.site.register(Category, CategoryAdmin)
34 admin.site.register(Product, ProductAdmin)
35 admin.site.register(ProductContent, ProductContentAdmin)
```

Settings:



Results:

All clear, no errors found

Python Validation

products/forms.py

```
8 class CategoryForm(forms.ModelForm):
9     """ Form for adding categories """
10     class Meta:
11         model = Category
12         fields = [
13             'name', 'friendly_name',
14         ]
15
16
17 class ProductForm(forms.ModelForm):
18     """ Form for adding products """
19     categories = Category.objects.all()
20     friendly_names = [(c.id, c.get_friendly_name()) for c in categories]
21
22     category = forms.MultipleChoiceField(
23         choices=friendly_names,
24         widget=forms.CheckboxSelectMultiple,
25     )
26
27     class Meta:
28         model = Product
29         fields = [
30             'name', 'description_short',
31             'description',
32             'category',
33             'price', 'duration', 'perks',
34             'image', 'alt_atr', 'online_onsite',
35         ]
36
37     def __init__(self, *args, **kwargs):
38         super().__init__(*args, **kwargs)
```

Settings:



Results:

All clear, no errors found

Python Validation

products/models.py

```
3
4 class Category(models.Model):
5     """ Model for products categories """
6     class Meta:
7         verbose_name_plural = 'Categories'
8
9     name = models.CharField(max_length=254)
10    friendly_name = models.CharField(max_length=254, null=True, blank=True)
11
12    def __str__(self):
13        return self.name
14
15    def get_friendly_name(self):
16        return self.friendly_name
17
18
19 class Product(models.Model):
20     """ Model for products description """
21     COMBINED = 'combined'
22     ONSITE = 'onsite'
23     ONLINE = 'online'
24     OPTIONAL = 'optional'
25
26     ONLINE_ONSITE_CHOICES = [
27         (COMBINED, 'Combined'),
28         (ONSITE, 'Onsite'),
29         (ONLINE, 'Online'),
30         (OPTIONAL, 'Optional'),
31     ]
32
33     name = models.CharField(max_length=254)
```

Settings:



Results:

All clear, no errors found

Python Validation

products/urls.py

```
3
4 urlpatterns = [
5     path('', views.all_products, name='products'),
6     path(
7         '<int:product_id>/',
8         views.product_detail,
9         name='product_detail'
10    ),
11    path(
12        'products/manage/',
13        views.product_management,
14        name='product_management'
15    ),
16    path(
17        'add/',
18        views.add_product,
19        name='add_product'
20    ),
21    path(
22        'addcontent/<int:product_id>/',
23        views.add_product_content,
24        name='add_product_content'
25    ),
26    path(
27        'edit/<int:product_id>/',
28        views.edit_product,
29        name='edit_product'
30    ),
31    path(
32        'delete/<int:product_id>/',
33        views.delete_product,
34        name='delete_product'
```

Settings:



Results:

All clear, no errors found

Python Validation

products/views.py

```
15
16 def all_products(request):
17     """ A view to return products page """
18
19     products = Product.objects.all()
20     product_contents = ProductContent.objects.all()
21     categories = Category.objects.all()
22     query = None
23
24     if 'category' in request.GET:
25         category_name = request.GET['category']
26         if category_name:
27             products = products.filter(category__name=category_name)
28             product_contents = product_contents.filter(
29                 product__category__name=category_name
30             )
31
32     if request.GET:
33         if 'q' in request.GET:
34             query = request.GET['q']
35             if not query:
36                 messages.error(
37                     request, "You didn't enter any search criteria!"
38                 )
39             else:
40                 queries = Q(name__icontains=query) | \
41                     Q(description__icontains=query)
42                 products = products.filter(queries)
43
44         if not products.exists():
45             return redirect(reverse('products'))
46
```

Settings:



Results:

All clear, no errors found

Python Validation

products/widgets.py

```
1 from django.forms.widgets import ClearableFileInput
2 from django.utils.translation import gettext_lazy as _
3
4
5 class CustomClearableFileInput(ClearableFileInput):
6     """ Widget that fixed the less good looking image input button in forms """
7     clear_checkbox_label = _('Remove')
8     initial_text = _('Current Image')
9     input_text = _('.')
10    template_name = (
11        'products/custom_widget_templates/custom_clearable_file_input.html'
12    )
13
```

Settings:

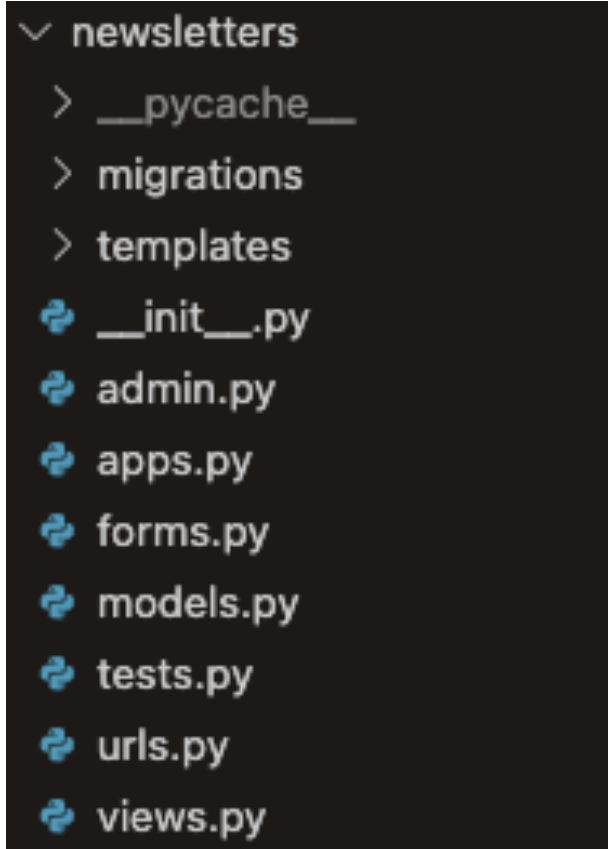


Results:

All clear, no errors found

Python Validation

App: newsletters



Python Validation

newsletters/admin.py

```
1 from django.contrib import admin
2 from .models import Newsletter
3
4
5 class NewsletterAdmin(admin.ModelAdmin):
6     """ Sets up admin panel for newsletters """
7     list_display = ('title', 'newsletter_category', 'created_at')
8     list_filter = ('newsletter_category', 'created_at')
9     search_fields = ('title', 'content')
10    date_hierarchy = 'created_at'
11
12
13 admin.site.register(Newsletter, NewsletterAdmin)
14
```

Settings:



Results:

All clear, no errors found

Python Validation

newsletters/forms.py

```
1 from django import forms
2 from .models import Newsletter
3
4
5 class NewsletterForm(forms.ModelForm):
6     """ Form for sending out lewsletters to newsletter subscribers """
7     class Meta:
8         model = Newsletter
9         fields = [
10             'title',
11             'content',
12             'newsletter_category',
13         ]
14 |
```

Settings:



Results:

All clear, no errors found

Python Validation

newsletters/models.py

```
1 from django.db import models
2
3
4 class Newsletter(models.Model):
5     """ Model for creating newsletters to subscribers """
6     NEWSLETTER_CATEGORIES = [
7         ('general', 'General'),
8         ('promotions', 'Promotions'),
9         ('product_updates', 'Product Updates'),
10    ]
11
12    title = models.CharField(max_length=254, blank=False, null=False)
13    content = models.TextField(blank=False, null=False)
14    newsletter_category = models.CharField(
15        max_length=20,
16        choices=NEWSLETTER_CATEGORIES,
17        default='general'
18    )
19    created_at = models.DateTimeField(auto_now_add=True)
20
21    def __str__(self):
22        return self.title
23
```

Settings:



Results:

All clear, no errors found

Python Validation

newsletters/urls.py

```
1 from django.urls import path
2 from . import views
3
4
5 urlpatterns = [
6     path('create/', views.create_newsletter, name='create_newsletter'),
7     path('', views.newsletters, name='newsletters'),
8 ]
9
```

Settings:



Results:

All clear, no errors found

Python Validation

newsletters/views.py

```
12 @staff_member_required
13 def newsletters(request):
14     """
15     A view for the administrator to see a
16     list of previous newsletters, categorized
17     """
18     newsletters_list = Newsletter.objects.all().order_by('-created_at')
19     if 'newsletter_category' in request.GET:
20         category = request.GET['newsletter_category']
21         if category:
22             newsletters_list = newsletters_list.filter(
23                 newsletter_category=category
24             )
25
26     template = 'newsletters/newsletters.html'
27     context = {
28         'newsletters_list': newsletters_list,
29     }
30
31     return render(request, template, context)
32
33
34 @staff_member_required
35 def create_newsletter(request):
36     """ View to create a newsletter """
37     if request.method == 'POST':
38         form = NewsletterForm(request.POST)
39         if form.is_valid():
40             newsletter = form.save()
41             user_profiles = UserProfile.objects.filter(
42                 newsletter_subscription=True
```

Settings:

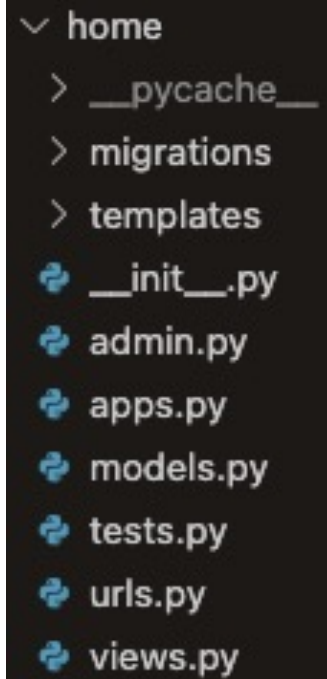


Results:

All clear, no errors found

Python Validation

App: home



```
▼ home
  > __pycache__
  > migrations
  > templates
  + __init__.py
  + admin.py
  + apps.py
  + models.py
  + tests.py
  + urls.py
  + views.py
```

A screenshot of a file explorer window showing the directory structure of a Django application named 'home'. The 'home' directory is expanded, revealing subdirectories for '__pycache__', 'migrations', and 'templates'. It also lists several Python files: '__init__.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', 'urls.py', and 'views.py'. Each file is preceded by a blue icon representing a Python file.

Python Validation

home/views.py

```
1 from django.shortcuts import render
2
3
4 def index(request):
5     """ A view to return home page """
6
7     return render(request, 'home/index.html')
8 |
```

Settings:



Results:

All clear, no errors found

Python Validation

newsletters/urls.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index, name='home'),
6 ]
7 |
```

Settings:

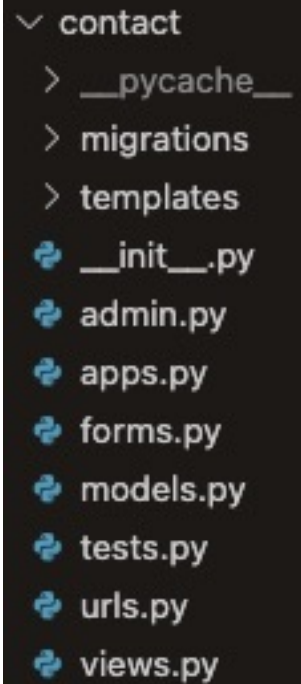


Results:

All clear, no errors found

Python Validation

App: contact



A screenshot of a file explorer window with a dark background. The 'contact' directory is expanded, showing its contents. The items are listed as follows:

- contact
 - > __pycache__
 - > migrations
 - > templates
 - 🔗 __init__.py
 - 🔗 admin.py
 - 🔗 apps.py
 - 🔗 forms.py
 - 🔗 models.py
 - 🔗 tests.py
 - 🔗 urls.py
 - 🔗 views.py

Python Validation

contact/admin.py

```
1 from django.contrib import admin
2 from .models import Contact
3
4
5 class ContactAdmin(admin.ModelAdmin):
6     """ Admin setup for contacts in the admin panel """
7     list_display = (
8         'name',
9         'phone',
10        'email',
11        'image',
12    )
13
14
15 admin.site.register(Contact)
16
```

Settings:



Results:

All clear, no errors found

Python Validation

contact/forms.py

```
1 from django import forms
2 from .models import Contact
3
4
5 class ContactForm(forms.ModelForm):
6     """ Form for adding a contact person """
7
8     class Meta:
9         model = Contact
10        fields = ['name', 'email', 'phone_number', 'image']
11
```

Settings:



Results:

All clear, no errors found

Python Validation

contact/models.py

```
1 from django.db import models
2
3
4 class Contact(models.Model):
5     """ Model for contact persons """
6     name = models.CharField(max_length=100, null=False, blank=True)
7     email = models.EmailField(null=False, blank=False)
8     phone_number = models.CharField(max_length=20, blank=False, null=False)
9     image = models.ImageField(null=False, blank=False)
10
11     def __str__(self):
12         return self.name
13
```

Settings:



Results:

All clear, no errors found

Python Validation

contact/urls.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.contact, name='contact'),
6     path(
7         'add_contact/',
8         views.add_contact,
9         name='add_contact'
10    ),
11    path(
12        'edit_contact/<int:contact_id>/',
13        views.edit_contact,
14        name='edit_contact'
15    ),
16    path(
17        'delete_contact/<int:contact_id>/',
18        views.delete_contact,
19        name='delete_contact'
20    ),
21 ]
22
```

Settings:



Results:

All clear, no errors found

Python Validation

contact/views.py

```
8
9 def contact(request):
10     """ A view to return contact page """
11     contacts = Contact.objects.all()
12
13     return render(request, 'contact/contact.html', {'contacts': contacts})
14
15
16 @staff_member_required
17 def add_contact(request):
18     """ Add contact view """
19     if not request.user.is_superuser:
20         messages.error(request, 'Sorry, only store owners can do that.')
21         return redirect(reverse('home'))
22     if request.method == 'POST':
23         form = ContactForm(request.POST, request.FILES)
24         if form.is_valid():
25             contact = form.save()
26             messages.success(
27                 request,
28                 f'You successfully added {contact.name}!'
29             )
30             return redirect('contact')
31         else:
32             messages.error(
33                 request,
34                 'Something went wrong, check if form is valid!'
35             )
36     else:
37         form = ContactForm()
38
39     template = 'contact/add_contact.html'
```

Settings:

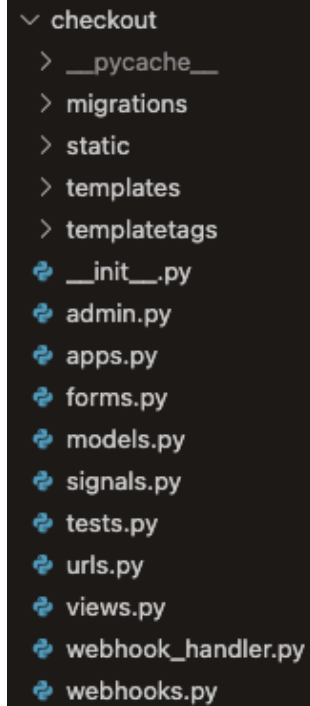


Results:

All clear, no errors found

Python Validation

App: checkout



```
▼ checkout
  > __pycache__
  > migrations
  > static
  > templates
  > templatetags
  + __init__.py
  + admin.py
  + apps.py
  + forms.py
  + models.py
  + signals.py
  + tests.py
  + urls.py
  + views.py
  + webhook_handler.py
  + webhooks.py
```

A screenshot of a file explorer showing the directory structure of a Django application named 'checkout'. The directory is expanded, revealing subdirectories and files. Subdirectories include '__pycache__', 'migrations', 'static', 'templates', and 'templatetags'. Files include '__init__.py', 'admin.py', 'apps.py', 'forms.py', 'models.py', 'signals.py', 'tests.py', 'urls.py', 'views.py', 'webhook_handler.py', and 'webhooks.py'. Each file is preceded by a small blue icon representing a Python file.

Python Validation

checkout/admin.py

```
4
5 class OrderLineItemInline(admin.TabularInline):
6     """ Sets up admin panel for products """
7     model = OrderLineItem
8     readonly_fields = ('product', 'quantity', 'lineitem_total')
9
10
11 class OrderAdmin(admin.ModelAdmin):
12     """ Sets up admin panel for user information """
13     inlines = (OrderLineItemInline,)
14
15     readonly_fields = (
16         'order_number',
17         'created',
18         'order_total',
19         'grand_total',
20         'tax',
21     )
22
23     fields = (
24         'order_number',
25         'created',
26         'user_profile',
27         'payment_option',
28         'invoice_ref',
29         'order_total',
30         'grand_total',
31     )
32
33     list_display = (
34         'order_number',
35         'payment_option',
```

Settings:



Results:

All clear, no errors found

Python Validation

checkout/forms.py

```
9
10 class CheckoutForm(forms.ModelForm):
11     """
12     Choices to pay with card or invoice.
13     if user chose invoice, they need to add invoice_ref to the order info
14     """
15     invoice_ref = forms.CharField(max_length=255, required=False)
16
17     class Meta:
18         model = Order
19         fields = (
20             'payment_option',
21             'invoice_ref',
22         )
23
24     def __init__(self, *args, **kwargs):
25         super().__init__(*args, **kwargs)
26
27         self.fields['payment_option'].choices = self.get_payment_options()
28
29         placeholders = {
30             'invoice_ref': 'Invoice Referens',
31         }
32
33         for field in self.fields:
34             if field in placeholders:
35                 if self.fields[field].required:
36                     placeholder = f'{placeholders[field]} *'
37                 else:
38                     placeholder = placeholders[field]
39                 self.fields[field].widget.attrs['placeholder'] = placeholder
40                 self.fields[field].widget.attrs['class'] = 'stripe-style-input'
```

Settings:



Results:

All clear, no errors found

Python Validation

checkout/models.py

```
14
15 ~ class Order(models.Model):
16     """ Model for the order that sets up database with order information """
17     INVOICE = 'invoice'
18     CARD = 'card'
19
20 ~     PAYMENT_OPTIONS = [
21         (INVOICE, 'Invoice'),
22         (CARD, 'Card'),
23     ]
24
25     order_number = models.CharField(max_length=10, unique=True)
26     user_profile = models.ForeignKey(
27         UserProfile,
28         on_delete=models.CASCADE,
29         related_name='orders'
30     )
31     created = models.DateTimeField(auto_now_add=True)
32     order_total = models.DecimalField(
33         max_digits=8,
34         decimal_places=2,
35         null=False,
36         blank=False
37     )
38     grand_total = models.DecimalField(
39         max_digits=8,
40         decimal_places=2,
41         null=False,
42         blank=False
43     )
44     tax = models.DecimalField(
45         max_digits=8
```

Settings:



Results:

All clear, no errors found

Python Validation

checkout/signals.py

```
1 from django.db.models.signals import post_save, post_delete
2 from django.dispatch import receiver
3
4 from .models import OrderLineItem
5
6
7 @receiver(post_save, sender=OrderLineItem)
8 def update_on_save(sender, instance, created, **kwargs):
9     """ Update order total on lineitem update/create """
10    instance.order.update_total()
11
12
13 @receiver(post_delete, sender=OrderLineItem)
14 def update_on_delete(sender, instance, **kwargs):
15     """ Update order total on lineitem delete """
16    instance.order.update_total()
17
```

Settings:



Results:

All clear, no errors found

Python Validation

checkout/urls.py

```
1 from django.urls import path
2 from . import views
3 from .webhooks import webhook
4
5 urlpatterns = [
6     path('', views.checkout, name='checkout'),
7     path(
8         'order/manage/',
9         views.order_management,
10        name='order_management'
11    ),
12    path(
13        'delete/<int:order_id>/',
14        views.delete_order,
15        name='delete_order'
16    ),
17    path(
18        'checkout_success/<order_number>',
19        views.checkout_success,
20        name='checkout_success'
21    ),
22    path(
23        'cache_checkout_data/',
24        views.cache_checkout_data,
25        name='cache_checkout_data'
26    ),
27 ]
28
```

Settings:



Results:

All clear, no errors found

Python Validation

checkout/views.py

```
27
28 @require_POST
29 def cache_checkout_data(request):
30     """ Function for caching the checkout data """
31     try:
32         pid = request.POST.get('client_secret').split('_secret')[0]
33         stripe.api_key = settings.STRIPE_SECRET_KEY
34         stripe.PaymentIntent.modify(pid, metadata={
35             'bag': json.dumps(request.session.get('bag', {})),
36         })
37         return HttpResponse(status=200)
38     except Exception as e:
39         messages.error(request, 'Sorry, your payment cannot be \
40             processed right now. Please try again later.')
41         return HttpResponse(content=e, status=400)
42
43
44 stripe.api_key = settings.STRIPE_SECRET_KEY
45
46
47 @login_required
48 def checkout(request):
49     """ Function to handle invoice or card payment """
50     stripe_public_key = settings.STRIPE_PUBLIC_KEY
51     stripe_secret_key = settings.STRIPE_SECRET_KEY
52
53     bag_context = bag_contents(request)
54     bag_items = bag_context['bag_items']
55     total = bag_context['total']
56     grand_total = bag_context['grand_total']
57     tax = bag_context['tax']
58
```

Settings:



Results:

All clear, no errors found

Python Validation

checkout/webhook_handler.py

```
1 from django.http import HttpResponse
2
3
4 # Webhooks created but not activated with the payment functionality
5 class StripeWH_Handler:
6     """Handle Stripe webhooks"""
7
8     def __init__(self, request):
9         self.request = request
10
11     def handle_event(self, event):
12         """ Handle a generic/unknown/unexpected webhook event """
13         return HttpResponse(
14             content=f'Unhandled webhook received: {event["type"]}',
15             status=200)
16
17     def handle_payment_intent_succeeded(self, event):
18         """ Handle a payment_intent.succeeded webhook from Stripe """
19         intent = event.data.object
20         return HttpResponse(
21             content=f'Webhook received: {event["type"]}',
22             status=200)
23
24     def handle_payment_intent_payment_failed(self, event):
25         """ Handle a payment_intent.payment_failed webhook from Stripe """
26         return HttpResponse(
27             content=f'Webhook received: {event["type"]}',
28             status=200)
29
```

Settings:



Results:

All clear, no errors found

Python Validation

checkout/webhooks.py

```
1 from django.conf import settings
2 from django.http import HttpResponse
3 from django.views.decorators.http import require_POST
4 from django.views.decorators.csrf import csrf_exempt
5
6 from checkout.webhook_handler import StripeWH_Handler
7
8 import stripe
9
10
11 # Webhooks created but not activated with the payment functionality
12 @require_POST
13 @csrf_exempt
14 def webhook(request):
15     """Listen for webhooks from Stripe"""
16     wh_secret = settings.STRIPE_WH_SECRET
17     stripe.api_key = settings.STRIPE_SECRET_KEY
18
19     payload = request.body
20     sig_header = request.META['HTTP_STRIPE_SIGNATURE']
21     event = None
22
23     try:
24         event = stripe.Webhook.construct_event(
25             payload, sig_header, wh_secret
26         )
27     except ValueError as e:
28         return HttpResponse(status=400)
29     except stripe.error.SignatureVerificationError as e:
30         return HttpResponse(status=400)
31     except Exception as e:
```

Settings:

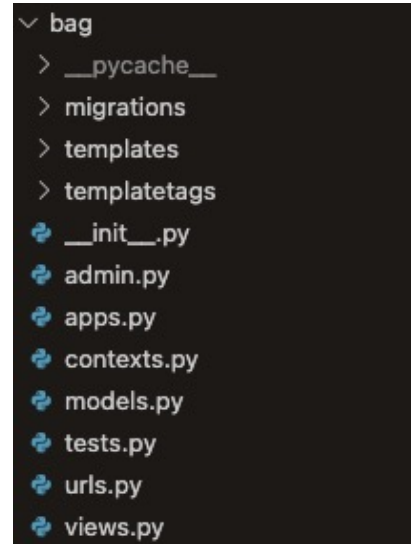


Results:

All clear, no errors found

Python Validation

App: bag



```
✓ bag
  > __pycache__
  > migrations
  > templates
  > templatetags
  + __init__.py
  + admin.py
  + apps.py
  + contexts.py
  + models.py
  + tests.py
  + urls.py
  + views.py
```

A screenshot of a file explorer showing the directory structure of a Django application named 'bag'. The structure includes a 'bag' directory containing subdirectories like '__pycache__', 'migrations', 'templates', and 'templatetags', as well as Python files such as '__init__.py', 'admin.py', 'apps.py', 'contexts.py', 'models.py', 'tests.py', 'urls.py', and 'views.py'.

Python Validation

bag/contexts.py

```
8
9 def bag_contents(request):
10     """ A context processor to use bag information across all templates """
11     bag_items = []
12     total = Decimal(0)
13     product_count = 0
14     grand_total = Decimal(0)
15     bag = request.session.get('bag', {})
16
17     for item_id, item_data in bag.items():
18         if isinstance(item_data, int):
19             product = get_object_or_404(Product, pk=item_id)
20             order_total = item_data * product.price
21             grand_total += order_total
22             product_count += item_data
23             bag_items.append({
24                 'item_id': item_id,
25                 'quantity': item_data,
26                 'product': product,
27                 'item_total': order_total,
28             })
29             total += order_total
30         else:
31             product = get_object_or_404(Product, pk=item_id)
32             order_total = item_data['quantity'] * product.price
33             grand_total += order_total
34             product_count += item_data['quantity']
35             bag_items.append({
36                 'item_id': item_id,
37                 'quantity': item_data['quantity'],
38                 'product': product,
39                 'item_total': order_total,
40             })
41             total += order_total
```

Settings:



Results:

All clear, no errors found

Python Validation

bag/urls.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.view_bag, name='view_bag'),
6     path('add/<item_id>', views.add_to_bag, name='add_to_bag'),
7     path('edit/<item_id>', views.edit_bag, name='edit_bag'),
8     path('delete/<item_id>', views.delete_item, name='delete_item'),
9 ]
10
```

Settings:



Results:

All clear, no errors found

Python Validation

bag/views.py

```
1 from django.shortcuts import (
2     render, redirect, reverse, HttpResponseRedirect, get_object_or_404
3 )
4 from django.contrib import messages
5
6 from products.models import Product
7
8
9 def view_bag(request):
10     """ A view to return the shopping bag """
11
12     return render(request, 'bag/bag.html')
13
14 def add_to_bag(request, item_id):
15     """ View for adding items to bag """
16     product = get_object_or_404(Product, pk=item_id)
17     quantity = int(request.POST.get('quantity', 1))
18     redirect_url = request.POST.get('redirect_url')
19     bag = request.session.get('bag', {})
20
21     if item_id in bag:
22         bag[item_id]['quantity'] += quantity
23         messages.success(
24             request,
25             f'Updated {product.name} quantity to {bag[item_id]["quantity"]}')
26     else:
27         bag[item_id] = {'quantity': quantity}
28         messages.success(request, f'Added {product.name} to your bag')
29
30     request.session['bag'] = bag
31     return redirect(redirect_url)
```

Settings:



Results:

All clear, no errors found