







Python Testing




Bag

<pre>1 from django.apps import AppConfig 2 3 4 class BagConfig(AppConfig): 5 default_auto_field = 'django.db.models.BigAutoField' 6 name = 'bag' 7</pre>	<p>Settings:</p> <p>  </p> <p>Results:</p> <p>All clear, no errors found</p>
--	--




apps.py

<pre>1 from django.conf import settings 2 from decimal import Decimal 3 from django.shortcuts import get_object_or_404 4 from products.models import Product 5 6 7 def bag_contents(request): 8 9 bag_items = [] 10 total = 0 11 product_count = 0 12 bag = request.session.get('bag', {}) 13 14 for item_id, item_data in bag.items():</pre>	<p>Settings:</p> <p>  </p> <p>Results:</p> <p>All clear, no errors found</p>
---	--

contexts.py



<pre>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('adjust/<item_id>', views.adjust_bag, name='adjust_bag'), 8 path('remove/<item_id>', views.remove_from_bag, name='remove_from_bag'), 9] 10</pre>	<p>Settings:</p> <p>  </p> <p>Results:</p> <p>All clear, no errors found</p>
---	--

urls.py




<pre>70 71 if quantity > 0: 72 bag[item_id] = quantity 73 else: 74 bag.pop(item_id) 75 76 request.session['bag'] = bag 77 return redirect(reverse('view_bag')) 78 79 def remove_from_bag(request, item_id): 80 """Remove item from the bag""" 81 82 try:</pre>	<p>Settings:</p> <p>  </p> <p>Results:</p> <p>All clear, no errors found</p>
---	--

views.py

checkout

<pre>1 from django.contrib import admin 2 from .models import Order, OrderLineItem 3 4 5 # Register your models here. 6 class OrderLineItemAdminInline(admin.TabularInline): 7 model = OrderLineItem 8 readonly_fields = ('lineitem_total',) 9 10 11 class OrderAdmin(admin.ModelAdmin): 12 inlines = (OrderLineItemAdminInline,) 13</pre>	<p>Settings:</p> <p>  </p> <p>Results:</p> <p>All clear, no errors found</p>
--	--


admin.py

<pre>1 from django.apps import AppConfig 2 3 4 class CheckoutConfig(AppConfig): 5 default_auto_field = 'django.db.models.BigAutoField' 6 name = 'checkout' 7 8 def ready(self): 9 import checkout.signals 10</pre>	<p>Settings:</p> <p>  </p> <p>Results:</p> <p>All clear, no errors found</p>
--	--

apps.py

```
5 ~ class OrderForm(forms.ModelForm):
6 ~     class Meta:
7 ~         model = Order
8 ~         fields = (
9 ~             'full_name', 'email', 'phone_number', 'street_address1',
10 ~             'street_address2', 'town_or_city', 'county', 'postcode', 'country')
11 ~
12 ~     def __init__(self, *args, **kwargs):
13 ~         """
14 ~         Add placeholders and classes, remove auto-generated
15 ~         labels and set autofocus on first field
16 ~         """
17 ~         super().__init__(*args, **kwargs)
```

Settings:

 ☒ 


Results:

All clear, no errors found

forms.py

```
63 ~
64 ~     Override original save method to set the order number if it hasn't
65 ~     been set already
66 ~     """
67 ~     if not self.order_number:
68 ~         self.order_number = self._generate_order_number()
69 ~     super().save(*args, **kwargs)
70 ~
71 ~     def __str__(self):
72 ~         return self.order_number
73 ~
74 ~ class OrderLineItem(models.Model):
75 ~     """Relates to specific orders"""
```

Settings:

 ☒ 

Results:

All clear, no errors found

models.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 ~
11 ~     instance.order.update_total()
12 ~
13 ~
```

Settings:

 ☒ 

Results:

All clear, no errors found

signals.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 ~         'checkout_success/<order_number>', views.checkout_success,
9 ~         name='checkout_success'),
10 ~     path(
11 ~         'cache_checkout_data/', views.cache_checkout_data,
12 ~         name='cache_checkout_data'),
13 ~     path('webhook/', views.webhook, name='webhook')]
14 ~
```

Settings:

 ☒ 

Results:

All clear, no errors found

urls.py

```
192 ~
193 ~     if request.user.is_authenticated:
194 ~         profile = UserProfile.objects.get(user_id=request.user.id)
195 ~         # Attach the user's profile to the order
196 ~         order.user_profile = profile
197 ~         order.save()
198 ~
199 ~         # save address to profile if checkbox is ticked
200 ~         if save_info:
201 ~             address_data = {
202 ~                 'street_address1': order.street_address1,
203 ~                 'street_address2': order.street_address2,
204 ~                 'town_or_city': order.town_or_city,
```

Settings:

 ☒ 



Results:

All clear, no errors found

views.py

```
132 ~         order=order,
133 ~         product=product,
134 ~         quantity=item_data,
135 ~     )
136 ~     order_line_item.save()
137 ~
138 ~     else:
139 ~         for size, quantity in item_data[
140 ~             'items_by_size'].items():
141 ~             order_line_item = OrderLineItem(
142 ~                 order=order,
143 ~                 product=product,
144 ~                 quantity=quantity,
```

Settings:

 ☒ 



Results:

All clear, no errors found

webhook_handler.py

```
21 ~ sig_header = request.META['HTTP_STRIPE_SIGNATURE']
22 ~ event = None
23 ~
24 ~ try:
25 ~     event = stripe.Webhook.construct_event(
26 ~         payload, sig_header, wh_secret)
27 ~ except ValueError as e:
28 ~     # Invalid payload
29 ~     return HttpResponse(status=400)
30 ~ except stripe.error.SignatureVerificationError as e:
31 ~     # Invalid signature
32 ~     return HttpResponse(status=400)
33 ~ except Exception as e:
```

Settings:

 ☒ 

Results:



All clear, no errors found

webhooks.py

Home

```
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

apps.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

urls.py

```
1 from django.shortcuts import render
2 from products.models import Category
3
4
5 # Create your views here.
6 def index(request):
7     categories = Category.objects.all()
8
9     context = {
10         'categories': categories,
11     }
12     return render(request, 'home/index.html', context)
13
```

Settings:

 ☒ 

Results:

All clear, no errors found



views.py

Products

```
1 from django.contrib import admin
2 from .models import Product, Category, OtherImages
3
4
5 # Register your models here.
6 class ProductAdmin(admin.ModelAdmin):
7     list_display = (
8         'sku',
9         'name',
10        'category',
11        'price',
12        'main_image',
13    )

```

Settings:

 ☒ 

Results:

All clear, no errors found

admin.py

```
1 from django.apps import AppConfig
2
3
4 class ProductsConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'products'
7
```

Settings:

 ☒ 

Results:



All clear, no errors found

apps.py

```
1 from django import forms
2 from .widgets import CustomClearableFileInput
3 from .models import Product, Category, OtherImages
4
5
6 class ProductForm(forms.ModelForm):
7
8     class Meta:
9         model = Product
10        fields = "__all__"
11
12    main_image = forms.ImageField(
13        label="", required=False, widget=CustomClearableFileInput)

```

Settings:

 ☒ 

Results:

All clear, no errors found

forms.py

```
10 name = models.CharField(
11     max_length=100, null=False, unique=True, blank=False,)
12 friendly_name = models.CharField(max_length=150, null=True, blank=True,)
13 image = models.ImageField(null=True, blank=True)
14
15 def __str__(self):
16     return self.name
17
18 def get_friendly_name(self):
19     return self.friendly_name
20
21
22 class Product(models.Model):
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

models.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.all_products, name='products'),
6     path('<int:product_id>', views.product_detail, name='product_detail'),
7     path('add/', views.add_product, name='add_product'),
8     path('add_image/', views.add_product_image, name='add_product_image'),
9     path('edit/<int:product_id>', views.edit_product, name='edit_product'),
10     path(
11         'delete/<int:product_id>', views.delete_product,
12         name='delete_product'),
13     path(
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

urls.py

```
172 @login_required
173 def delete_product(request, product_id):
174     """ Delete a product """
175
176     if not request.user.is_superuser:
177         messages.error(
178             request, 'Sorry, only store owners can delete a product')
179         return redirect(reverse('home'))
180
181     product = get_object_or_404(Product, pk=product_id)
182     product.delete()
183     messages.success(request, 'Product deleted')
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

views.py

```
1 from django.forms.widgets import ClearableFileInput
2 from django.utils.translation import gettext_lazy as _
3
4
5 class CustomClearableFileInput(ClearableFileInput):
6     clear_checkbox_label = _('Remove')
7     initial_text = _('Main Image')
8     input_text = _('')
9     template_name = (
10         'products/custom_widget_templates/custom_clearable_file_input.html')
11 |
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

widgets.py

Profiles

```
1 from django.contrib import admin
2 from .models import Addresses, UserProfile
3
4
5 # Register your models here.
6 class UserProfileAdmin(admin.ModelAdmin):
7     list_display = (
8         'user',
9         'phone_number',
10     )
11
12
13 class AddressesAdmin(admin.ModelAdmin):
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

admin.py

```
1 from django.apps import AppConfig
2
3
4 class ProfilesConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'profiles'
7 |
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

apps.py

```
7 model = Addresses
8 fields = (
9     'street_address1', 'street_address2', 'town_or_city', 'county',
10     'postcode', 'country', 'default_address')
11 widgets = {
12     'default_address': forms.CheckboxInput()
13 }
14
15 def __init__(self, *args, **kwargs):
16     """
17     Add placeholders and classes, remove auto-generated
18     labels and set autofocus on first field
19 """
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

forms.py

```
20
21 @receiver(post_save, sender=User)
22 def create_or_update_user_profile(sender, instance, created, **kwargs):
23     """ Create or update user profile"""
24     if created:
25         UserProfile.objects.create(user=instance)
26         instance.userprofile.save()
27
28
29 class Addresses(models.Model):
30     """
31     Model for addresses
32     """
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

models.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.profile, name='profile'),
6     path('edit/<int:address_id>', views.edit_address, name='edit_address'),
7     path(
8         'delete/<int:address_id>', views.delete_address,
9         name='delete_address'),
10    path(
11        'order_history/<order_number>', views.order_history,
12        name='order_history'),
13 ]
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

urls.py

```
55         return redirect(reverse('profile'))
56     else:
57         messages.error(
58             request, 'Unable to update address. Please try again')
59
60     form = AddressForm(instance=address)
61
62     template = 'profiles/edit_address.html'
63     context = {
64         'address': address,
65         'form': form,
66     }
67
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

views.py

Unnie_style

```
1 """Unnie_style URL Configuration
2
3 The 'urlpatterns' list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.2/topics/http/urls/
5 Examples:
6 Function views
7 1. Add an import: from my_app import views
8 2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10 1. Add an import: from other_app.views import Home
11 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13 1. Import the include() function: from django.urls import include, path
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

urls.py

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

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

views.py

Wishlist

```
1 from django.contrib import admin
2 from .models import Wishlist
3
4
5 # Register your models here.
6 class WishlistAdmin(admin.ModelAdmin):
7     list_display = (
8         'user_profile',
9         'products',
10    )
11
12
13 admin.site.register(Wishlist, WishlistAdmin)
```

Settings:

🌙 ☒ ☀️

Results:

All clear, no errors found

admin.py

```
1 from django.apps import AppConfig
2
3
4 class WishlistConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'wishlist'
7
```

Settings:




Results:

All clear, no errors found

apps.py

```
1 from django.db import models
2 from products.models import Product
3 from profiles.models import UserProfile
4
5
6 # Create your models here.
7 class Wishlist(models.Model):
8     """Wishlist model"""
9     products = models.ForeignKey(
10         Product, null=True, blank=True, on_delete=models.CASCADE)
11     user_profile = models.ForeignKey(
12         UserProfile, null=True, blank=True, on_delete=models.CASCADE)
13
```

Settings:




Results:

All clear, no errors found

models.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.wishlist, name='wishlist'),
6     path('delete/<int:wishlist_id>', views.delete_wish, name='delete_wish'),
7 ]
8
```

Settings:




Results:

All clear, no errors found

urls.py

```
1 from django.shortcuts import render, redirect, reverse, get_object_or_404
2 from django.contrib.auth.decorators import login_required
3 from .models import Wishlist
4 from profiles.models import UserProfile
5
6
7 # Create your views here.
8 @login_required
9 def wishlist(request):
10     """ Display the user's wishlist"""
11
12     profile = get_object_or_404(UserProfile, user=request.user)
13     wishlist = Wishlist.objects.filter(user_profile=profile)
```

Settings:



Results:

All clear, no errors found

views.py

```
1 from django.conf import settings
2 from storages.backends.s3boto3 import S3Boto3Storage
3
4
5 class StaticStorage(S3Boto3Storage):
6     location = settings.STATICFILES_LOCATION
7
8
9 class MediaStorage(S3Boto3Storage):
10     location = settings.MEDIAFILES_LOCATION
11 |
```

Settings:

Results:

All clear, no errors found

custom_storages.py