

Settings.py

The screenshot shows a dark-themed application window titled "CI Python Linter". In the top left corner, there is a logo for "code institute". The main area displays a code editor with a snippet of Python code for a Django settings file. The code includes imports for os, dj_database_url, and Path, along with configurations for DEBUG, ALLOWED_HOSTS, CSRF_TRUSTED_ORIGINS, SECRET_KEY, and INSTALLED_APPS. The code editor has line numbers from 1 to 34 on the left. On the right side, there is a sidebar with "Settings:" and a toggle switch, and a "Results:" section stating "All clear, no errors found".

```
1 import os
2 import dj_database_url
3 from pathlib import Path
4
5 if os.path.isfile("env.py"):
6     import env
7
8 BASE_DIR = Path(__file__).resolve().parent.parent
9
10 # Security settings
11 DEBUG = True
12 ALLOWED_HOSTS = [
13     "localhost",
14     "127.0.0.1",
15     "artera-d6829bf39792.herokuapp.com",
16 ]
17
18 CSRF_TRUSTED_ORIGINS = [
19     "https://artera-d6829bf39792.herokuapp.com",
20     "https://localhost",
21     "https://127.0.0.1",
22 ]
23
24 SECRET_KEY = os.getenv("SECRET_KEY", "")
25
26 # Application definition
27 INSTALLED_APPS = [
28     "django.contrib.admin",
29     "django.contrib.auth",
30     "django.contrib.contenttypes",
31     "django.contrib.sessions",
32     "django.contrib.messages",
33     "django.contrib.staticfiles",
34     "django.contrib.sites",
```

Bag. Context.py

The screenshot shows the CI Python Linter interface. On the left is a dark-themed code editor window with line numbers and syntax highlighting for Python code. The code defines a `bag_contents` function that retrieves items from a session and calculates their total price. On the right, there's a sidebar with "Settings:" and a light/dark mode toggle. Below it is a "Results:" section stating "All clear, no errors found".

```
1  from decimal import Decimal
2  from django.conf import settings
3  from django.shortcuts import get_object_or_404
4  from shop.models import Artwork
5
6
7  def bag_contents(request):
8      """Works with the simple bag structure from your views"""
9
10     bag_items = []
11     total = Decimal("0")
12     product_count = 0
13     bag = request.session.get("bag", {})
14
15     for item_id, item_data in bag.items():
16         artwork = get_object_or_404(Artwork, pk=item_id)
17         unit_price = Decimal(getattr(artwork, "price", 0) or 0)
18
19         # If item has sizes
20         if (
21             isinstance(item_data, dict)
22             and "sizes" in item_data
23             and isinstance(item_data["sizes"], dict)
24         ):
25             for size, quantity in item_data["sizes"].items():
26                 qty = int(quantity) if quantity else 1
27                 line_total = unit_price * qty
28                 total += line_total
29                 product_count += qty
30                 bag_items.append(
31                     {
32                         "item_id": item_id,
33                         "quantity": qty,
34                         "artwork": artwork,
```

Bag. Apps.py

The screenshot shows a dark-themed interface for a CI Python Linter. At the top left is the Code Institute logo. To its right is the title "CI Python Linter". On the far right are three icons: a moon (dark mode), a toggle switch, and a sun (light mode).

The main area is divided into two sections: "Settings" on the left and "Results" on the right. The "Settings" section contains a single line of code:

```
from django.apps import AppConfig
```


The "Results" section displays the message: "All clear, no errors found".

```
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

The screenshot shows a dark-themed interface for a CI Python Linter. At the top left is the Code Institute logo. To its right is the title "CI Python Linter". On the far right are three small icons: a crescent moon, a toggle switch, and a sun.

The main area contains a code editor window displaying a Django URL configuration file:

```
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(
8         "remove/<int:item_id>/", views.remove_from_bag, name="remove_from_bag",
9     ),
10 ]
11
```

To the right of the code editor is a "Results" section with the message "All clear, no errors found".

Bag. Views.py

The screenshot shows a dark-themed interface for a CI Python Linter. On the left, there's a code editor window with the 'code institute' logo at the top. The code editor displays a Python file named 'Views.py' with the following content:

```
1  from django.shortcuts import (
2      render,
3      redirect,
4      reverse,
5      HttpResponseRedirect,
6      get_object_or_404,
7  )
8  from django.contrib import messages
9  from django.views.decorators.http import require_POST
10 from shop.models import Artwork
11
12
13 def view_bag(request):
14     """Render the bag contents page"""
15     return render(request, "bag/bag.html")
16
17
18 @require_POST
19 def add_to_bag(request, item_id):
20     """
21     Add the specified artwork to the shopping bag.
22     Shape enforced to match context:
23     | bag[item_id] = {"sizes": {size_code: 1, ...}}
24     Quantities are fixed at 1 (no increments).
25     """
26     artwork = get_object_or_404(Artwork, pk=item_id)
27     redirect_url = request.POST.get("redirect_url") or reverse("view_bag")
28
29     size = request.POST.get("product_size")
30     if not size:
31         messages.error(request, "Please choose a size.")
32         return redirect(redirect_url)
33
34     bag = request.session.get("bag", {})
```

To the right of the code editor is a sidebar with the following sections:

- Settings:** Includes a gear icon, a toggle switch, and a sun icon.
- Results:** Displays the message "All clear, no errors found".

Checkout. Admin.py



The image shows a screenshot of the CI Python Linter application. On the left, there is a dark-themed code editor window with the "code institute" logo at the top. The code editor displays a Python file named "admin.py". The code defines two classes: "OrderItemAdminInline" and "OrderAdmin". The "OrderItemAdminInline" class is a TabularInline for the "OrderItem" model, with extra=0 and readonly_fields set to "('lineitem_total',)". The "OrderAdmin" class is a ModelAdmin for the "Order" model, with inlines set to "OrderItemAdminInline". It also defines readonly_fields and fields for various order-related fields. The code editor has line numbers from 1 to 34.

CI Python Linter

```
1  from django.contrib import admin
2  from .models import Order, OrderItem
3
4  |
5  class OrderItemAdminInline(admin.TabularInline):
6      model = OrderItem
7      extra = 0
8      readonly_fields = ('lineitem_total',)
9
10 @admin.register(Order)
11 class OrderAdmin(admin.ModelAdmin):
12     inlines = (OrderItemAdminInline,)
13
14     readonly_fields = (
15         'order_number',
16         'date',
17         'delivery_cost',
18         'order_total',
19         'grand_total',
20     )
21
22     fields = (
23         'order_number',
24         'date',
25
26         'full_name',
27         'email',
28         'phone_number',
29
30         'country',
31         'postcode',
32         'town_or_city',
33         'street_address1',
```

Settings:

Results:

All clear, no errors found

Checkout. Apps.py

The screenshot shows a dark-themed interface for a CI Python Linter. On the left, there's a code editor window with the Code Institute logo at the top. The code editor displays the following Python code:

```
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
```

To the right of the code editor is a panel titled "CI Python Linter". It contains two sections: "Settings:" and "Results:". The "Settings:" section includes a toggle switch between a moon and a sun icon. The "Results:" section displays the message "All clear, no errors found".

Checkout.Forms.py

The screenshot shows a dark-themed interface for a CI Python Linter. On the left, there's a code editor window with the 'code institute' logo at the top. The code editor displays a Python file named 'Checkout.Forms.py'. The code defines a form class 'OrderForm' that inherits from 'ModelForm' and specifies fields for a 'Order' model. It includes placeholder labels for address fields and an __init__ method with detailed documentation. On the right, there's a sidebar titled 'Results' which states 'All clear, no errors found'. Above the results, there are 'Settings' options for light/dark mode and a toggle switch.

```
1  from django import forms
2  from .models import Order
3
4
5  class OrderForm(forms.ModelForm):
6      class Meta:
7          model = Order
8          fields = (
9              "full_name",
10             "email",
11             "phone_number",
12             "street_address1",
13             "street_address2",
14             "town_or_city",
15             "postcode",
16             "country",
17             "county",
18         )
19         labels = {
20             "street_address1": "Delivery address",
21             "street_address2": "Spare address",
22         }
23
24     def __init__(self, *args, **kwargs):
25         """
26             Add placeholders and classes, remove auto-generated labels,
27             and set autofocus on the first field.
28         """
29         super().__init__(*args, **kwargs)
30
31         placeholders = {
32             "full_name": "Full Name",
33             "email": "Email Address",
34             "phone_number": "Phone Number",
```

Checkout. Models.py



The image shows a screenshot of a software application titled "CI Python Linter". On the left, there is a code editor window with the "code institute" logo at the top. The code editor displays a Python file named "Models.py" with line numbers from 93 to 126. The code defines a class with various fields and methods, including a save method that calculates a total price and updates it. On the right, there is a results panel with a "Settings" section containing a toggle switch and a brightness icon, and a "Results" section stating "All clear, no errors found".

```
93     null=False,
94     blank=False,
95     on_delete=models.CASCADE,
96     related_name="lineitems",
97   )
98 artwork = models.ForeignKey(
99   Artwork,
100  null=False,
101  blank=False,
102  on_delete=models.PROTECT
103 )
104 size = models.CharField(max_length=5, null=True, blank=True)
105 quantity = models.IntegerField(null=False, blank=False, default=0)
106 lineitem_total = models.DecimalField(
107   max_digits=10,
108   decimal_places=2,
109   null=False,
110   blank=False,
111   editable=False
112 )
113
114 def save(self, *args, **kwargs):
115   """
116   Set the line item total (unit price * qty).
117   Snapshot the current artwork price to avoid future price drift.
118   """
119   self.lineitem_total = (
120     self.artwork.price * self.quantity
121     ).quantize(Decimal("0.01"))
122   super().save(*args, **kwargs)
123
124 def __str__(self):
125   return f'Artwork {self.artwork.id} on order {self.order.order_number}'
```

Checkout. Signals.py

The screenshot shows a dark-themed application window titled "CI Python Linter". In the top left corner, there is a logo for "code institute". The main area contains a code editor with the following Python code:

```
1 from django.db.models.signals import post_save, post_delete
2 from django.dispatch import receiver
3
4 from .models import OrderItem
5
6
7 @receiver(post_save, sender=OrderItem)
8 def update_on_save(sender, instance, created, **kwargs):
9     """
10     Update order total on lineitem create/update.
11     """
12     instance.order.update_total()
13
14
15 @receiver(post_delete, sender=OrderItem)
16 def update_on_delete(sender, instance, **kwargs):
17     """
18     Update order total on lineitem delete.
19     """
20     instance.order.update_total()
21
```

To the right of the code editor, there is a "Settings" section with a moon and sun icon for theme selection and a toggle switch. Below that is a "Results" section with the message "All clear, no errors found".

Checkout.Urls.py

The screenshot shows a dark-themed interface for a CI Python Linter. In the top left corner, the "code institute" logo is displayed. To its right, the text "CI Python Linter" is written. On the far right, there is a "Settings:" section containing three icons: a gear, a toggle switch, and a sun-like icon. Below this is a "Results:" section with the message "All clear, no errors found". The main area of the interface is a code editor window. The code shown is:

```
1 from django.urls import path
2 from . import views
3 from . import webhooks
4 from .webhooks import webhook
5
6 urlpatterns = [
7     path('', views.checkout, name='checkout'),
8     path('success<order_number>/',
9         views.checkout_success, name='checkout_success'),
10    path('webhook/', webhooks.webhook, name='stripe_webhook'),
11    path('cache_checkout_data/',
12        views.cache_checkout_data, name='cache_checkout_data'),
13    path('wh/', webhook, name='webhook'),
14]
15|
```

Checkout.Views.py

The screenshot shows a dark-themed interface for a CI Python Linter. At the top left is the Code Institute logo. To the right, the title "CI Python Linter" is displayed. On the far right, there are "Settings:" controls for night mode, a toggle switch, and a brightness icon. Below these settings, the "Results:" section displays the message "All clear, no errors found". The main area contains a code editor window showing a Python file named "Checkout.Views.py". The code imports various Django modules and defines a function "send_order_confirmation" that sends an email with a rendered template.

```
1 from django.shortcuts import (
2     render,
3     redirect,
4     reverse,
5     get_object_or_404,
6     HttpResponseRedirect,
7 )
8 from django.views.decorators.http import require_POST
9 from django_countries import countries
10 from django.contrib import messages
11 from django.conf import settings
12 from django.core.mail import EmailMultiAlternatives
13 from django.template.loader import render_to_string
14 from django.contrib.sites.models import Site
15
16 from .forms import OrderForm
17 from .models import Order, OrderItem
18 from shop.models import Artwork
19 from bag.contexts import bag_contents
20 from profiles.models import Profile
21
22 import stripe
23 import json
24
25
26 def send_order_confirmation(order):
27     """Send a confirmation email for a completed order."""
28     site = Site.objects.get_current()
29     ctx = {
30         "order": order,
31         "site": site,
32     }
33
34     subject = render_to_string(
```

Checkout. Webhook_handler.py

The screenshot shows a dark-themed interface for a CI Python Linter. In the top left corner, there is a logo for "code institute". The main area displays a block of Python code for a webhook handler. The code is annotated with line numbers from 1 to 34. The code itself is as follows:

```
1  from django.http import HttpResponseRedirect
2  from .models import Order, OrderItem
3  from shop.models import Artwork
4
5  import stripe
6  import json
7
8
9  class StripeWH_Handler:
10     """Handle Stripe webhooks"""
11
12     def __init__(self, request):
13         self.request = request
14
15     def handle_event(self, event):
16         """
17             Handle a generic/unknown/unexpected webhook event
18         """
19         intent = event.data.object
20         pid = intent.id
21         bag = intent.metadata.bag
22         save_info = intent.metadata.save_info
23
24         billing_details = intent.charges.data[0].billing_details
25         shipping_details = intent.shipping
26         grand_total = round(intent.data.charges[0].amount / 100, 2)
27
28         # Clean data in the shipping details
29         for field, value in shipping_details.address.items():
30             if value == "":
31                 shipping_details.address[field] = None
32         order_exists = False
33         try:
34             order = Order.objects.get(
```

In the top right, the title "CI Python Linter" is displayed. To the right of the code editor, there is a sidebar titled "Settings:" which includes a light/dark mode toggle and a "Results:" section stating "All clear, no errors found".

Checkout. Webhooks.py

The screenshot shows a dark-themed interface for a CI Python Linter. In the top left corner, there's a logo for "code institute". The main area contains a code editor window with the following Python code:

```
1 from django.http import HttpResponse, HttpResponseRedirect
2 from django.conf import settings
3 from django.views.decorators.http import require_POST
4 from django.views.decorators.csrf import csrf_exempt
5 from .webhook_handler import StripeWH_Handler # same app -> relative import
6 import stripe
7
8 @require_POST
9 @csrf_exempt
10 def webhook(request):
11     """Listen for webhooks from Stripe"""
12     stripe.api_key = settings.STRIPE_SECRET_KEY
13     wh_secret = settings.STRIPE_WH_SECRET
14
15     payload = request.body
16     sig_header = request.META.get("HTTP_STRIPE_SIGNATURE", "")
17     if not sig_header:
18         return HttpResponseRedirect("Missing Stripe signature")
19
20     try:
21         event = stripe.Webhook.construct_event(payload, sig_header, wh_secret)
22     except ValueError:
23         # Invalid payload
24         return HttpResponseRedirect("Invalid payload")
25     except stripe.error.SignatureVerificationError:
26         # Invalid signature
27         return HttpResponseRedirect("Invalid signature")
28
29     # Connect to handler
30     handler = StripeWH_Handler(request)
31     event_map = {
32         "payment_intent.succeeded":
33             handler.handle_payment_intent_succeeded,
```

To the right of the code editor, there are sections for "Settings" and "Results". The "Settings" section includes a toggle switch for light/dark mode and a sun icon. The "Results" section displays the message "All clear, no errors found".

Home. Admin.py

The screenshot shows a dark-themed interface for a CI Python Linter. On the left, there's a code editor window with the Code Institute logo at the top. The code editor displays the following Python code:

```
1 from django.contrib import admin
2 from .models import SearchDocument
3
4
5 @admin.register(SearchDocument)
6 class SearchDocumentAdmin(admin.ModelAdmin):
7     list_display = ("title", "url")
8     search_fields = (
9         "title",
10        "description",
11        "content",
12        "extra_text",
13        "url",
14        "categories__name",
15        "categories__friendly_name"
16    )
17    filter_horizontal = ("categories",)
18
```

To the right of the code editor is a results panel titled "CI Python Linter". It contains a "Settings:" section with a light/dark mode toggle and a "Results:" section stating "All clear, no errors found".

Home. Apps.py

The screenshot shows the CI Python Linter interface. On the left, there's a dark-themed code editor window with the Code Institute logo at the top. The code in the editor is:

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

To the right of the code editor is the "CI Python Linter" title. Below it are two sections: "Settings:" and "Results:". The "Settings:" section contains a toggle switch between a dark mode icon and a light mode icon. The "Results:" section displays the message "All clear, no errors found".

Home. Models.py

The screenshot shows a dark-themed interface for a Python linter. On the left, there's a code editor window with the title "code institute". It contains a snippet of Django model code:

```
1 from django.db import models
2 from shop.models import category
3
4
5 class SearchDocument(models.Model):
6     title = models.CharField(max_length=255)
7     content = models.TextField()
8     description = models.TextField(blank=True, null=True)
9     extra_text = models.TextField(blank=True, null=True)
10    url = models.CharField(max_length=500)
11    categories = models.ManyToManyField(
12        Category,
13        blank=True,
14        related_name="search_docs"
15    )
16
17    def __str__(self):
18        return self.title
19
```

To the right of the code editor is a panel titled "CI Python Linter". It includes a "Settings:" section with a moon and sun icon for themes, and a "Results:" section stating "All clear, no errors found".

Home.Urls.py



The screenshot shows a dark-themed interface for a CI Python Linter. On the left, there's a code editor window with the Code Institute logo at the top. The code editor displays the following Python code:

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.homepage, name='home'),
6     path('artwork-search/', views.artwork_search, name='artwork_search'),
7 ]
8 |
```

To the right of the code editor is a results panel titled "CI Python Linter". It contains two sections: "Settings:" and "Results:". The "Settings:" section includes a toggle switch and a sun icon. The "Results:" section displays the message "All clear, no errors found".

Home. Views.py

The screenshot shows a dark-themed application window titled "CI Python Linter". In the top left corner, there is a logo for "code institute". The main area contains a code editor with the following Python code:

```
1 from django.shortcuts import render, redirect
2 from django.urls import reverse
3 from urllib.parse import urlencode
4 from django.contrib import messages
5
6
7 def homepage(request):
8     return render(request, 'home/home.html')
9
10
11 def artwork_search(request):
12     q = (request.GET.get("q") or "").strip()
13     if not q:
14         messages.error(request, "You didn't enter any search criteria!")
15         return redirect(reverse('all_artworks'))
16
17     params = {'q': q}
18     if 'category' in request.GET:
19         params['category'] = request.GET['category']
20
21     return redirect(f"{reverse('all_artworks')}?{urlencode(params)}")
22
```

To the right of the code editor, there is a "Settings" section with a light/dark mode toggle switch. Below the code editor, the "Results" section displays the message: "All clear, no errors found".

Profiles. Apps.py

The screenshot shows a dark-themed interface for a CI Python linter. On the left is a code editor window with the Code Institute logo at the top. The code editor displays the following Python code:

```
1 from django.apps import AppConfig
2
3
4 class ProfilesConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'profiles'
7
8     def ready(self):
9         import profiles.signals
10
```

To the right of the code editor is a status bar with the text "CI Python Linter". Below the status bar are two sections: "Settings:" and "Results:". The "Settings:" section contains a moon and sun icon with a toggle switch, indicating a dark mode setting. The "Results:" section displays the message "All clear, no errors found".

Profiles. Forms.py

The screenshot shows a dark-themed interface for a CI Python Linter. On the left, there's a logo for 'code institute' and a code editor window displaying a Python file named 'Profiles. Forms.py'. The code defines a 'ProfileForm' class that inherits from 'ModelForm'. It includes fields for full name, phone number, address line delivery, address line living, city, postal code, and country, each with specific CSS classes applied via 'attrs'. Labels for these fields are also defined. On the right, there's a 'Settings' section with a light/dark mode toggle and a 'Results' section stating 'All clear, no errors found'.

```
1  from django import forms
2  from .models import Profile
3  from allauth.account.forms import SignupForm
4
5
6  class ProfileForm(forms.ModelForm):
7      class Meta:
8          model = Profile
9          fields = [
10              'full_name', 'phone_number', 'address_line_delivery',
11              'address_line_living', 'city', 'postal_code', 'country'
12          ]
13          widgets = {
14              'full_name': forms.TextInput(attrs={'class': 'form-control'}),
15              'phone_number': forms.TextInput(attrs={'class': 'form-control'}),
16              'address_line_delivery': forms.TextInput(
17                  attrs={'class': 'form-control'}
18              ),
19              'address_line_living': forms.TextInput(
20                  attrs={'class': 'form-control'}
21              ),
22              'city': forms.TextInput(attrs={'class': 'form-control'}),
23              'postal_code': forms.TextInput(attrs={'class': 'form-control'}),
24              'country': forms.TextInput(attrs={'class': 'form-control'})
25          }
26          labels = {
27              'full_name': 'Full Name',
28              'phone_number': 'Phone Number',
29              'address_line_delivery': 'Delivery Address',
30              'address_line_living': 'Living Address',
31              'city': 'city',
32              'postal_code': 'Postal Code',
33              'country': 'Country',
34          }
```

Profiles. Models.py

The screenshot shows the CI Python Linter interface. At the top left is the Code Institute logo. To the right is the title "CI Python Linter". On the far right are "Settings:" controls for dark mode, a toggle switch, and a brightness icon. Below these are "Results:" and a message "All clear, no errors found". The main area contains a code editor with a dark theme, displaying a Django models.py file. The code defines a Profile model with various fields like full_name, phone_number, address_line_delivery, address_line_living, city, postal_code, country, created_at, updated_at, and avatar. A line number 24 is highlighted.

```
1 from django.db import models
2 from django.conf import settings
3 from cloudinary.models import CloudinaryField
4
5
6 class Profile(models.Model):
7     user = models.OneToOneField(
8         settings.AUTH_USER_MODEL,
9         on_delete=models.CASCADE
10    )
11    full_name = models.CharField(max_length=100, blank=True)
12    phone_number = models.CharField(max_length=20, blank=True)
13    address_line_delivery = models.CharField(max_length=255, blank=True)
14    address_line_living = models.CharField(max_length=255, blank=True)
15    city = models.CharField(max_length=100, blank=True)
16    postal_code = models.CharField(max_length=20, blank=True)
17    country = models.CharField(max_length=100, blank=True)
18    created_at = models.DateTimeField(auto_now_add=True)
19    updated_at = models.DateTimeField(auto_now=True)
20    avatar = CloudinaryField('image', blank=True, null=True, folder='avatars')
21
22 def __str__(self):
23     return f"{self.user.username}'s profile"
24
```

Profiles. Signals.py

The screenshot shows a dark-themed interface for a CI Python Linter. In the top left, the Code Institute logo is visible. The title "CI Python Linter" is centered at the top. On the right, there's a "Settings:" section with a toggle switch and a sun/moon icon. Below it is a "Results:" section displaying the message "All clear, no errors found". The main area contains a code editor with the following Python code:

```
1 from django.db.models.signals import post_save
2 from django.dispatch import receiver
3 from django.conf import settings
4 from .models import Profile
5 from allauth.account.signals import email_confirmed
6 from allauth.account.models import EmailAddress
7
8
9 @receiver(post_save, sender=settings.AUTH_USER_MODEL)
10 def create_user_profile(sender, instance, created, **kwargs):
11     if created:
12         Profile.objects.create(user=instance)
13
14
15 @receiver(email_confirmed)
16 def make_confirmed_email_primary(request, email_address, **kwargs):
17     """When a user confirms a new email, make it primary"""
18     user = email_address.user
19     # Make this the only primary
20     EmailAddress.objects.filter(user=user).update(primary=False)
21     email_address.primary = True
22     email_address.save(update_fields=['primary'])
23
24     user.email = email_address.email
25     user.save(update_fields=['email'])
26 |
```

Profiles.Urls.py

The screenshot shows the CI Python Linter interface. On the left is a code editor with the following content:

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.profile, name='profile'),
6     path('profile/avatar/upload/', views.avatar_upload, name='avatar_upload'),
7     path('profile/avatar/remove/', views.avatar_remove, name='avatar_remove'),
8     path('library/', views.library, name='library'),
9     path("library/download/<slug:order_number>/<int:lineitem_id>/",
10          views.download_artwork, name="download_artwork"),
11 ]
12 ]
```

The code editor has a dark theme with syntax highlighting. Lines 1 through 11 are visible, and line 12 is highlighted in purple.

To the right of the code editor are two sections: "Settings:" and "Results:". The "Settings:" section contains a toggle switch between night mode (dark background) and light mode (white background). The "Results:" section displays the message "All clear, no errors found".

Profiles. Views.py

The screenshot shows a software interface for continuous integration (CI) Python linter. On the left, there is a code editor window displaying a Python script. The script includes imports from django.contrib.auth.decorators, django.shortcuts, django.http, django.db, django.db.models, django.contrib.messages, allauth.account.models, django.http, and clouddinary.uploader. It defines a profile function that handles POST requests to update a user's profile. The code uses Django's ORM and form validation. On the right, there is a settings panel with a dark mode toggle and a results panel indicating "All clear, no errors found".

```
1 from django.contrib.auth.decorators import login_required
2 from django.shortcuts import render, redirect, get_object_or_404
3 from django.http import HttpResponseRedirect
4 from django.db import transaction
5 from django.db.models import Prefetch, Count
6 from django.contrib import messages
7 from allauth.account.models import EmailAddress
8 from django.http import HttpResponseRedirect
9 import cloudinary.uploader
10
11
12 from checkout.models import OrderItem, Order
13 from .forms import ProfileForm
14 from .models import Profile
15 from services.models import ArtworkRequest
16
17
18 @login_required
19 def profile(request):
20     profile, _ = Profile.objects.get_or_create(user=request.user)
21
22     if request.method == "POST":
23         form = ProfileForm(request.POST, instance=profile)
24         new_email = (request.POST.get("email") or "").strip()
25
26         if form.is_valid():
27             with transaction.atomic():
28                 form.save()
29                 if (
30                     new_email
31                     and new_email.lower()
32                     != (request.user.email or "").lower()
33                 ):
34                     # Create or reuse EmailAddress for the new email
```

Services. Admin.py

The screenshot shows a dark-themed interface for a CI Python Linter. On the left, there's a logo for "code institute". The main area displays a block of Python code for "admin.py". The code defines two ModelAdmin classes: "ArtworkRequestAdmin" and "OfferAdmin", both derived from "admin.ModelAdmin". The "ArtworkRequestAdmin" class has fields for "id", "title", "user", "status", and "created_at". It also includes filters for "status" and "created_at", search fields for "title", "description", and "user.username", and actions for "mark_in_review", "mark_accepted", and "mark_rejected". The "OfferAdmin" class has fields for "id", "title", "full_name", "email", and "created_at". It includes filters for "title", "full_name", "email", and "description", and search fields for "title", "full_name", "email", and "description". The code is numbered from 1 to 18.

```
1  from django.contrib import admin
2  from .models import ArtworkRequest, Offer
3
4
5  @admin.register(ArtworkRequest)
6  class ArtworkRequestAdmin(admin.ModelAdmin):
7      list_display = ("id", "title", "user", "status", "created_at")
8      list_filter = ("status", "created_at")
9      search_fields = ("title", "description", "user__username")
10     list_editable = ("status",)
11     actions = ["mark_in_review", "mark_accepted", "mark_rejected"]
12
13
14  @admin.register(Offer)
15  class OfferAdmin(admin.ModelAdmin):
16      list_display = ("id", "title", "full_name", "email", "created_at")
17      search_fields = ("title", "full_name", "email", "description")
18
```

CI Python Linter

Settings:

Results:

All clear, no errors found

Services. Apps.py

The screenshot shows a dark-themed interface for a CI Python Linter. On the left, there's a code editor window with the Code Institute logo at the top. The code in the editor is:

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

To the right of the code editor is a panel titled "CI Python Linter". It contains two sections: "Settings:" and "Results:". The "Settings:" section includes a toggle switch and a brightness icon. The "Results:" section displays the message "All clear, no errors found".

Services. Forms.py

The screenshot shows a dark-themed interface for a CI Python Linter. At the top left is the Code Institute logo. To the right, the title "CI Python Linter" is displayed. On the far right, there are "Settings:" controls for brightness and a toggle switch. Below these are "Results:" and a message stating "All clear, no errors found". The main area contains a code editor window with a dark background. The code shown is:

```
1 from django import forms
2 from .models import ArtworkRequest, Offer
3
4
5 class ArtworkRequestForm(forms.ModelForm):
6     class Meta:
7         model = ArtworkRequest
8         fields = ["title", "description", "ref_image", "budget_cents"]
9         widgets = {
10             "title": forms.TextInput(attrs={"class": "form-control"}),
11             "description": forms.Textarea(
12                 attrs={"class": "form-control", "rows": 4}
13             ),
14             "budget_cents": forms.NumberInput(
15                 attrs={"class": "form-control", "min": 0}
16             ),
17         }
18
19     def __init__(self, *args, **kwargs):
20         super().__init__(*args, **kwargs)
21
22         # Automatically add * to required fields
23         for name, field in self.fields.items():
24             if field.required:
25                 field.label = f"{field.label} *"
26
27
28 class OfferForm(forms.ModelForm):
29     class Meta:
30         model = Offer
31         fields = [
32             "full_name",
33             "email",
34             "phone_number",
```

Services. Models.py

The screenshot shows a dark-themed interface for a CI Python Linter. At the top left is the Code Institute logo. The main area contains a code editor with a snippet of Python code for a Django model named `ArtworkRequest`. The code defines fields like `user`, `title`, `description`, `ref_image`, `budget_cents`, and `status`. The code editor has line numbers from 1 to 34. On the right side, there's a sidebar titled "Settings" with a light/dark mode toggle and a "Results" section stating "All clear, no errors found".

```
1  from django.db import models
2  from django.conf import settings
3  from cloudinary.models import CloudinaryField
4
5
6  class ArtworkRequest(models.Model):
7      STATUS = [
8          ("in_review", "In review"),
9          ("accepted", "Accepted"),
10         ("rejected", "Rejected"),
11     ]
12    user = models.ForeignKey(
13        settings.AUTH_USER_MODEL,
14        on_delete=models.CASCADE,
15        related_name="service_requests"
16    )
17    title = models.CharField(max_length=120, blank=False)
18    description = models.TextField(blank=False, max_length=1000)
19    ref_image = CloudinaryField('image', folder='requests', blank=True)
20    budget_cents = models.DecimalField(
21        max_digits=10,
22        decimal_places=2,
23        null=True,
24        blank=True
25    )
26    status = models.CharField(
27        max_length=20,
28        choices=STATUS,
29        default="in_review"
30    )
31    created_at = models.DateTimeField(auto_now_add=True)
32
33
34  class Offer(models.Model):
```

Services.Urls.py

The screenshot shows a CI Python Linter interface. On the left, there's a dark-themed code editor window with the 'code institute' logo at the top. The code editor displays a Python file named 'Services.Urls.py' with the following content:

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('request', views.artwork_request, name='request'),
6     path("requests/int:pk/", views.request_detail, name="request_detail"),
7     path("requests/int:pk/edit/", views.request_edit, name="request_edit"),
8     path(
9         "requests/int:pk/delete/",
10        views.request_delete,
11        name="request_delete"
12    ),
13    path('offer', views.offer_request, name='offer'),
14    path(
15        "offer/success/int:offer_id/",
16        views.offer_success,
17        name="offer_success"
18    ),
19]
20
21 ]
```

To the right of the code editor is a light-colored sidebar with the title 'CI Python Linter'. It contains two sections: 'Settings:' and 'Results:'. The 'Settings:' section includes a dark mode toggle switch and a brightness slider. The 'Results:' section displays the message 'All clear, no errors found'.

Services. Views.py

The screenshot shows a dark-themed interface for a CI Python Linter. At the top left is the Code Institute logo. To the right, the title "CI Python Linter" is displayed. The main area contains a code editor with a snippet of Django Python code. The code handles artwork requests, including validation and saving to the database. The code editor has line numbers from 1 to 34. On the right side of the interface, there's a sidebar titled "Settings:" which includes a toggle switch and a sun/moon icon. Below it is a section titled "Results:" with the message "All clear, no errors found".

```
1  from django.shortcuts import render, redirect, get_object_or_404
2  from django.contrib.auth.decorators import login_required
3  from django.contrib import messages
4  from .forms import ArtworkRequestForm, OfferForm
5  from .models import ArtworkRequest
6  from .models import Offer
7
8
9  @login_required
10 def artwork_request(request):
11     if request.method == "POST":
12         form = ArtworkRequestForm(request.POST, request.FILES)
13         if form.is_valid():
14             obj = form.save(commit=False)
15             obj.user = request.user
16             obj.save()
17             messages.success(request, "Request submitted for review.")
18             return redirect("request_detail", pk=obj.pk)
19             messages.error(request, "Please fix the errors below.")
20     else:
21         form = ArtworkRequestForm()
22
23     return render(request, "services/request_form.html", {"form": form})
24
25
26 @login_required
27 def request_detail(request, pk):
28     obj = get_object_or_404(ArtworkRequest, pk=pk, user=request.user)
29     return render(
30         request,
31         "services/request_detail.html", {"request_obj": obj}
32     )
33
34
```

Shop. Admin.py

The screenshot shows a dark-themed interface for a CI Python Linter. In the top left, the "code institute" logo is displayed. To the right, the title "CI Python Linter" is centered. On the far right, there are three small icons: a moon (dark mode), a toggle switch, and a sun (light mode). Below these are two sections: "Settings:" and "Results:". The "Settings:" section contains the same three icons. The "Results:" section displays the message "All clear, no errors found". The main area of the interface is a code editor window containing the following Python code:

```
1 from django.contrib import admin
2 from .models import Artwork, Category
3
4
5 @admin.register(Artwork)
6 class ArtworkAdmin(admin.ModelAdmin):
7     list_display = ("name", "category", "sku", "price",)
8     list_filter = ("category",)
9     search_fields = ("name", "sku", "description")
10    autocomplete_fields = ("category",)
11
12
13 @admin.register(Category)
14 class CategoryAdmin(admin.ModelAdmin):
15     list_display = (
16         'friendly_name',
17         'name',
18     )
19     search_fields = ("name", "friendly_name")
20
21
22 @admin.display(description="sizes")
23 def sizes(self, obj):
24     return "S, M, L"
25
```

Shop. Apps.py

The screenshot shows a dark-themed user interface for a CI Python Linter. In the top left corner, the "code institute" logo is displayed. To the right of the logo, the text "CI Python Linter" is written. The main area contains a code editor window with the following Python code:

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

To the right of the code editor, there is a sidebar titled "Settings:" which includes a toggle switch and a brightness icon. Below the settings is a section titled "Results:" with the message "All clear, no errors found".

Shop. Models.py

The image shows a screenshot of a software application titled "CI Python Linter". On the left, there is a dark-themed code editor window with the "code institute" logo at the top. The code editor displays a Python file named "Shop. Models.py". The code defines a "Category" model with attributes like "name" and "friendly_name", and a "calc_size_price" function. On the right, there is a results panel titled "Results:" which states "All clear, no errors found". Above the results panel are "Settings:" controls for theme and brightness.

```
1 from django.db import models
2 from decimal import Decimal
3 from django.utils import timezone
4
5
6 SIZE_CHOICES = [
7     ("S", "640x959"),
8     ("M", "1280x1917"),
9     ("L", "1920x2876"),
10 ]
11
12 SIZE_SURCHARGE = {
13     "S": Decimal("0"),
14     "M": Decimal("5"),
15     "L": Decimal("10"),
16 }
17
18 def calc_size_price(base_price: Decimal, size_code: str) -> Decimal:
19     return (base_price + SIZE_SURCHARGE[size_code]).quantize(Decimal("0.01"))
20
21
22 class Category(models.Model):
23     class Meta:
24         verbose_name_plural = "Categories"
25
26         name = models.CharField(max_length=100)
27         friendly_name = models.CharField(max_length=100, null=True, blank=True)
28
29     def __str__(self):
30         return self.name
31
32     def get_friendly_name(self):
33         return self.friendly_name
```

Shop.Urls.py

The screenshot shows a dark-themed code editor interface for a Django project's `Shop.Urls.py` file. The editor displays the following code:

```
1 from django.urls import path
2 from . import views
3
4
5 urlpatterns = [
6     path('', views.all_artworks, name='all_artworks'),
7     path(
8         "category/<int:category_id>/",
9         views.all_artworks,
10        name="by_category"
11    ),
12    path("<int:artwork_id>", views.artwork_detail, name="detail"),
13    path("search/", views.artwork_search, name="artwork_search"),
14 ]
15
```

The interface includes a header with the **code institute** logo and the title **CI Python Linter**. On the right side, there are sections for **Settings:** (with a moon and sun icon) and **Results:** (showing the message **All clear, no errors found**). The code editor has a vertical scrollbar on the right.

Shop. Views.py

The screenshot shows the CI Python Linter application interface. At the top left is the Code Institute logo. The main area displays a Python file named `Shop. Views.py`. The code implements a view function `all_artworks` that handles both GET and POST requests. It filters artworks by category or search term and includes logic for handling size surcharges. On the right side, there are sections for **Settings** (with a dark mode toggle) and **Results** (which states "All clear, no errors found").

```
1  from django.shortcuts import render, redirect, reverse, get_object_or_404
2  from django.contrib import messages
3  from django.db.models import Q
4  from django.db.models.functions import Lower
5  from decimal import Decimal
6  from django.utils import timezone
7  from datetime import timedelta
8  from django.db.models.functions import Random
9
10 from .models import Artwork, Category, SIZE_CHOICES, SIZE_SURCHARGE
11
12
13 def all_artworks(request, category_id=None):
14     artworks = Artwork.objects.select_related("category").order_by(Random())
15     categories = Category.objects.order_by("name")
16
17     if category_id:
18         artworks = artworks.filter(category_id=category_id)
19     query = None
20     current_categories = None
21
22     if "category" in request.GET:
23         names = [
24             c.strip()
25             for c in request.GET["category"].split(",") if c.strip()]
26         artworks = artworks.filter(category_name__in=names)
27         current_categories = Category.objects.filter(name__in=names)
28
29     context = {
30         "artworks": artworks,
31         "search_term": query,
32         "current_categories": current_categories,
33         "SIZE_SURCHARGE": SIZE_SURCHARGE,
34         "categories": categories,
```