Python Validation

CI Python Linter

```python
from django.contrib import admin
from .models import About, CollaborateRequest
from django_summernote.admin import SummernoteModelAdmin


@admin.register(About)
class AboutAdmin(SummernoteModelAdmin):

    summernote_fields = ('content',)


@admin.register(CollaborateRequest)
class CollaborateRequestAdmin(admin.ModelAdmin):

    list_display = ('message', 'read',)

```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
from django.contrib import admin
from .models import About, CollaborateRequest
from django_summernote.admin import SummernoteModelAdmin


@admin.register(About)
class AboutAdmin(SummernoteModelAdmin):

    summernote_fields = ('content',)


@admin.register(CollaborateRequest)
class CollaborateRequestAdmin(admin.ModelAdmin):

    list_display = ('message', 'read',)

```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
from django.test import TestCase
from .forms import CollaborateForm


class TestCollaborateForm(TestCase):
    """Test suite for the CollaborateForm."""

    def test_form_is_valid(self):
        """Test for all fields when the form is valid."""
        form_data = {
            'name': 'Matt',
            'email': 'test@test.com',
            'message': 'Hello!'
        }
        form = CollaborateForm(form_data)
        print("Form errors:", form.errors)
        self.assertTrue(form.is_valid(), msg="Form is not valid")

    def test_name_is_required(self):
        """Test for the 'name' field when it is empty."""
        form_data = {
            'name': '',
            'email': 'test@test.com',
            'message': 'Hello!'
        }
        form = CollaborateForm(form_data)
        self.assertFalse(
            form.is_valid(),
            msg="Name was not provided, but the form is valid"
        )

```

Settings:

Results:

All clear, no errors found

code institute

CI Python Linter

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

Settings:

Results:

All clear, no errors found

```python
from django.shortcuts import render
from django.contrib import messages
from .models import About
from .forms import CollaborateForm


def about_me(request):
    """
    Renders the About page.
    Handles collaboration form submission and displays
    the most recent About section content.
    """
    if request.method == "POST":
        collaborate_form = CollaborateForm(data=request.POST)
        if collaborate_form.is_valid():
            collaborate_form.save()
            messages.add_message(
                request,
                messages.SUCCESS,
                "Collaboration request received! "
                "I endeavour to respond within 2 working days."
            )
    else:
        collaborate_form = CollaborateForm()

    about = About.objects.all().order_by('-updated_on').first()

    return render(
        request,
        "about/about.html",
        {
            "about": about,
            "collaborate_form": collaborate_form,
        },
    )

```

Settings:

Results:

All clear, no errors found

institute

```python
from django.contrib import admin
from django.utils.html import format_html
from django_summernote.admin import SummernoteModelAdmin
from .models import Post, Comment


@admin.register(Post)
class PostAdmin(SummernoteModelAdmin):
    """
    Admin interface for the Post model.
    """
    list_display = ('title', 'slug', 'status', 'created_on', 'image_thumbnail')
    search_fields = ['title', 'content']
    list_filter = ('status', 'created_on')
    prepopulated_fields = {'slug': ('title',)}
    summernote_fields = ('content',)

    def make_published(self, request, queryset):
        """
        Custom admin action to mark selected posts as published.
        """
        queryset.update(status='published')
    make_published.short_description = "Mark selected posts as Published"

    def image_thumbnail(self, obj):
        """
        Display a thumbnail for the featured image in the admin list.
        """
        if obj.featured_image:
            return format_html(
                '<img src="{}" width="10" height="10" alt="Thumbnail" />',
                obj.featured_image.url
            )
        return "No Image"


```

Settings:

Results:

All clear, no errors fou

```python
from django.apps import AppConfig


class BlogConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'blog'
```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
from django import forms
from django.core.exceptions import ValidationError
from .models import Comment, Post


def validate_file_size(file):
    """
    Custom validator to ensure file size does not exceed 10MB.
    """
    max_size = 10 * 1024 * 1024  # 10MB
    if file.size > max_size:
        readable_max_size = max_size // (1024 * 1024)
        raise ValidationError(
            f"File size too large. "
            f"Maximum size is {readable_max_size}MB."
        )


class CommentForm(forms.ModelForm):
    """
    Form for creating and managing comments.
    """
    class Meta:
        model = Comment
        fields = ('body',)


class PostForm(forms.ModelForm):
    """
    Form for creating and managing posts, including a custom validator
    for the featured image size.
    """
```

Settings:

Results:

All clear, no errors found

CI Python Linter

code institute

```python
from django.db import models
from django.contrib.auth.models import User
from cloudinary.models import CloudinaryField
from django.utils.text import slugify

STATUS = ((0, "Draft"), (1, "Published"))


class Post(models.Model):
    """
    Model representing a blog post.
    """
    title = models.CharField(max_length=200, unique=True)
    slug = models.SlugField(max_length=200, unique=True)
    author = models.ForeignKey(
        User, on_delete=models.CASCADE, related_name="blog_posts"
    )
    featured_image = CloudinaryField('image', default='placeholder')
    content = models.TextField()
    created_on = models.DateTimeField(auto_now_add=True)
    status = models.IntegerField(choices=STATUS, default=0)
    excerpt = models.TextField(blank=True)
    updated_on = models.DateTimeField(auto_now=True)

    class Meta:
        ordering = ["-created_on"]

    def __str__(self):
        return self.title

    def save(self, *args, **kwargs):
        """
```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
from django.test import TestCase
from .forms import CommentForm


class TestCommentForm(TestCase):
    """
    Tests for the CommentForm.
    """

    def test_form_is_valid(self):
        """
        Test that the form is valid when provided with valid data.
        """
        comment_form = CommentForm({'body': 'This is a great post'})
        self.assertTrue(comment_form.is_valid())

    def test_form_is_invalid(self):
        """
        Test that the form is invalid when the body field is empty.
        """
        comment_form = CommentForm({'body': ''})
        self.assertFalse(
            comment_form.is_valid(),
            msg="The form should not be valid with an empty body."
        )

```

Settings:

Results:

All clear, no errors fou

CI Python Linter

```python
from django.contrib.auth.models import User
from django.urls import reverse
from django.test import TestCase
from unittest.mock import patch
from .forms import CommentForm
from .models import Post


class TestBlogViews(TestCase):
    """
    Tests for blog views including rendering post details
    and comment submission.
    """

    def setUp(self):
        """
        Set up test data including a user and a blog post.
        """
        self.user = User.objects.create_superuser(
            username="myUsername",
            password="myPassword",
            email="test@test.com"
        )
        self.post = Post.objects.create(
            title="Blog title",
            author=self.user,
            slug="blog-title",
            excerpt="Blog excerpt",
            content="Blog content",
            status=1
        )

```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
from django.urls import path
from . import views

# Add URL patterns
urlpatterns = [
    path("submit/", views.submit_post, name="submit_post"),
    path("", views.PostList.as_view(), name="home"),
    path("home", views.PostList.as_view(), name="home"),
    path("<slug:slug>/", views.post_detail, name="post_detail"),
    path(
        "<slug:slug>/edit_comment/<int:comment_id>",
        views.comment_edit,
        name="comment_edit",
    ),
    path(
        "<slug:slug>/delete_comment/<int:comment_id>",
        views.comment_delete,
        name="comment_delete",
    ),
]

```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
from django.shortcuts import render, get_object_or_404, reverse, redirect
from django.contrib.auth.decorators import login_required
from django.utils.text import slugify
from django.views import generic
from django.contrib import messages
from django.http import HttpResponseRedirect
from .models import Post, Comment
from .forms import CommentForm, PostForm
from django.core.exceptions import ValidationError
from cloudinary.exceptions import Error


# Create your views here.
class PostList(generic.ListView):
    queryset = Post.objects.filter(status=1)
    template_name = "blog/index.html"
    paginate_by = 3


def post_detail(request, slug):
    """
    Display an individual :model:`blog.Post`.

    **Context**

    ``post``:
        An instance of :model:`blog.Post`.

    **Template**:
        :template:`blog/post_detail.html`
    """
    queryset = Post.objects.filter(status=1)
```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
"""
Django settings for diyblog project.

Generated by 'django-admin startproject' using Django 4.2.16.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""

from pathlib import Path
import os
import sys
import cloudinary
import cloudinary.uploader
import cloudinary.api
import dj_database_url
if os.path.isfile('env.py'):
    import env

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Template directory
TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY =
```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
"""
URL configuration for diyblog project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path("about/", include("about.urls"), name="about-urls"),
    path("accounts/", include("allauth.urls")),
    path('summernote/', include('django_summernote.urls')),
    path('admin/', admin.site.urls),
    path("", include("blog.urls"), name="blog-urls"),
]
```

Settings:

Results:

All clear, no errors found

code institute

CI Python Linter

```
"""
WSGI config for diyblog project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'diyblog.settings')

application = get_wsgi_application()

```

Settings:

Results:

All clear, no errors found

CI Python Linter

```python
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'diyblog.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Settings:

Results:

All clear, no errors fo