



CI Python Linter

Project urls.py

```
1  """little_learner_lab_logs 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
14     2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15  """
16  from django.contrib import admin
17  from django.urls import path, include
18
19  urlpatterns = [
20      path('admin/', admin.site.urls),
21      path('summernote/', include('django_summernote.urls')),
22      path('', include('logs.urls'), name='logs'),
23      path('profiles/', include('profiles.urls'), name='profiles'),
24      path('accounts/', include('allauth.urls')),
25  ]
26
27  handler404 = 'little_learner_lab_logs.views.handler404'
28
```

Setting



Results

All clear, no



CI Python Linter

Profiles models.py

```
1  ''' Profile Model Imports '''
2  from django.db import models
3  from django.contrib.auth.models import User
4  from django.db.models.signals import post_save
5  from django.dispatch import receiver
6  from django.core.exceptions import ObjectDoesNotExist
7  from cloudinary.models import CloudinaryField
8
9
10 # Profile Model
11 class Profile(models.Model):
12     '''
13     Profile model
14     Extends user model
15     '''
16     user = models.OneToOneField(
17         User,
18         on_delete=models.CASCADE,
19         blank=True,
20         null=True)
21     profile_image = CloudinaryField(
22         'image',
23         default='../default_profile_afxozd',
24         blank=True
25     )
26     brief_bio = models.TextField()
27
28     def __str__(self):
29         '''
30         To return the user's username object as a string
31         '''
32         return f'Profile for {self.user.username}'
33
34
35 @receiver(post_save, sender=User)
```

Settings



Results

All clear, no



CI Python Linter

Profiles forms.py

```
1 from django import forms
2 from django.forms import ModelForm, FileInput
3 from django.contrib.auth.models import User
4 from .models import Profile
5 from crispy_forms.helper import FormHelper
6 from crispy_forms.layout import Layout, Div, Field, Button, Submit, HTML
7 from crispy_forms.bootstrap import FormActions
8
9
10 class ProfileForm(ModelForm):
11     """
12     Create form class based on Profile model
13     Reference for the code in the link below:
14     https://github.com/Code-Institute-Submissions/andy-guttridge-song-mates
15     """
16     def __init__(self, *args, **kwargs):
17         """
18         Init form using crispy forms FormHelper for form layout
19         """
20         super(ProfileForm, self).__init__(*args, **kwargs)
21         self.helper = FormHelper(self)
22         self.helper.form_tag = False
23         self.helper.form_class = 'form-horizontal'
24         self.helper.field_class = 'col-md-12'
25         self.helper.layout = Layout(
26             HTML('<div class = "text-start">'),
27             Field('brief_bio', css_class='bkgnd-color-1 txt-color-2'),
28             Field('profile_image', css_class='bkgnd-color-1 txt-color-2'),
29             # Approach to using a HTML helper class to display an image from
30             # the data base in the form from
31             # https://stackoverflow.com/questions/21076248/imagefield-preview-in-crispy-forms-layout
32             Div(
33                 Div('Username: '),
34                 Div('Brief-Bio'),
35             ),
```

Settings



Results

All clear, no



CI Python Linter

Profiles views.py

```
1 from django.shortcuts import render, redirect, HttpResponseRedirect, get_object_or_404 # NOQA
2 from django.urls import reverse_lazy
3 from django.views import View
4 from django.utils.decorators import method_decorator
5 from django.template import RequestContext
6 from django.contrib import messages
7 from django.contrib.auth.decorators import login_required
8 from .forms import ProfileForm
9 from .models import Profile
10
11
12 class ProfileAccount(View):
13     """
14     User profile and account view.
15     Redirects to login page if user not authenticated.
16     Credits for the code:
17     https://github.com/Code-Institute-Submissions/andy-guttridge-song-mates
18     """
19     @method_decorator(login_required)
20     def get(self, request, *args, **kwargs):
21         if not Profile.objects.filter(user=request.user).exists():
22             profile = Profile(user=request.user)
23             profile.save()
24             # Retrieve profile from database, create form from it and
25             # render
26             profile_queryset = Profile.objects.filter(user=request.user)
27             get_object_or_404(profile_queryset)
28             profile = profile_queryset.first()
29             profile_form = ProfileForm(instance=profile)
30             return render(
31                 request,
32                 'edit_profile.html',
33                 {
34                     'profile': profile,
35                     'form': profile_form
```

Settings



Results

All clear, no



CI Python Linter

Profiles urls.py

```
1 from . import views
2 from django.urls import path
3
4 urlpatterns = [
5     path('edit-profile/', views.ProfileAccount.as_view(), name='edit_profile'),
6     path('user-delete/', views.UserDelete.as_view(), name='user_delete'),
7     path('update-profile/', views.UpdateProfile.as_view(),
8         name='update_profile'),
9 ]
10
```

Settings



Results

All clear, no



CI Python Linter

Logs models.py

```
1 from django.db import models
2 from django.contrib.auth.models import User
3 from clouinary.models import CloudinaryField
4 from django.urls import reverse
5 from django.template.defaultfilters import slugify
6
7
8 STATUS = ((0, "Draft"), (1, "Published"))
9
10
11 class Category(models.Model):
12     """
13     Category Model linked to each post
14     """
15     category_title = models.CharField(max_length=80, blank=False)
16     slug = models.SlugField(max_length=200, unique=True)
17
18     class Meta:
19         verbose_name = "category"
20         verbose_name_plural = "categories"
21
22     def __str__(self):
23         return self.category_title
24
25
26 class Post(models.Model):
27     """
28     Model for lab log posts
29     """
30     title = models.CharField(max_length=200, unique=True)
31     author = models.ForeignKey(
32         User,
33         on_delete=models.CASCADE,
34         related_name="log_posts",
35         blank=True,
```

Setting



Results

All clear, no



CI Python Linter

Logs forms.py

```
1 from django import forms
2 from django_summernote.widgets import SummernoteWidget
3 from .models import Post, Comment
4
5
6 class UploadFileForm(forms.Form):
7     """
8     To upload images
9     """
10    title = forms.CharField(max_length=100)
11    file = forms.FileField()
12
13    class Meta:
14        model = Post
15        fields = ['image']
16
17
18 class PostForm(forms.ModelForm):
19     """
20     Django form to add lab log posts
21     """
22    class Meta:
23        # Get post model and choose which fields to display
24        model = Post
25        fields = ('title', 'author', 'description', 'items_required', 'steps_to_perform', 'category',
26                  , 'image',) # noqa: E501
27
28    widgets = {
29        'title': forms.TextInput(attrs={
30            'class': 'form-control',
31            'placeholder': 'Title'}),
32        'author': forms.TextInput(attrs={
33            'class': 'form-control',
34            'value': '',
35            'id': 'author_field',
```

Settings



Results

All clear, no



CI Python Linter

Logs urls.py

```
1 from . import views
2 from django.urls import path
3 from .views import PostDetail, AllPosts, PostLike, SharedPostsByUsers, DeletePost # noqa: E501
4
5
6 urlpatterns = [
7     path('', views.PostList.as_view(), name='home'),
8     path('search/', views.PostSearch, name='search'),
9     path('lab_logs/', AllPosts.as_view(), name='lab-logs'),
10    path('my_page/', views.SharedPostsByUsers.as_view(), name='my-page'),
11    path('log_details/<slug:slug>', PostDetail.as_view(), name='log-details'),
12    path('like/<slug:slug>', views.PostLike.as_view(), name='post-like'),
13    path('add_logs/', views.CreatePost, name='add-logs'),
14    path('edit/<slug:slug>', views.EditPost, name='edit-logs'),
15    path('delete/<slug:slug>', DeletePost.as_view(), name='delete-logs'),
16 ]
17
```

Settings



Results

All clear, no



CI Python Linter

Logs views.py

```
1 from django.shortcuts import render, get_object_or_404, reverse, redirect
2 from django.contrib.auth.decorators import login_required
3 from django.views import generic, View
4 from django.http import HttpResponseRedirect
5 from django.contrib import messages
6 from django.utils.text import slugify
7 from django.contrib.auth.mixins import LoginRequiredMixin
8 from django.db.models import Q
9 from django.views.generic import ListView
10 from .models import Post
11 from .forms import PostForm, CommentForm, UploadFileForm, EditForm
12
13
14 class PostList(generic.ListView):
15     ''' Class to show list of posts'''
16     model = Post
17     queryset = Post.objects.filter(status=1).order_by('-created_on')
18     template_name = 'index.html'
19     paginate_by = 3
20
21
22 class AllPosts(generic.ListView):
23     """
24     to get all the lab log posts, and display 12 posts per page
25     """
26     model = Post
27     queryset = Post.objects.filter(status=1).order_by('created_on') # noqa: E501
28     template_name = 'lab_logs.html'
29     paginate_by = 12
30
31
32 class PostDetail(View):
33     ''' Class to show selected post in detail view '''
34     def get(self, request, slug, *args, **kwargs):
35         queryset = Post.objects.filter(status=1)
36         post = get_object_or_404(queryset, slug=slug)
```

Setting



Results

All clear, no