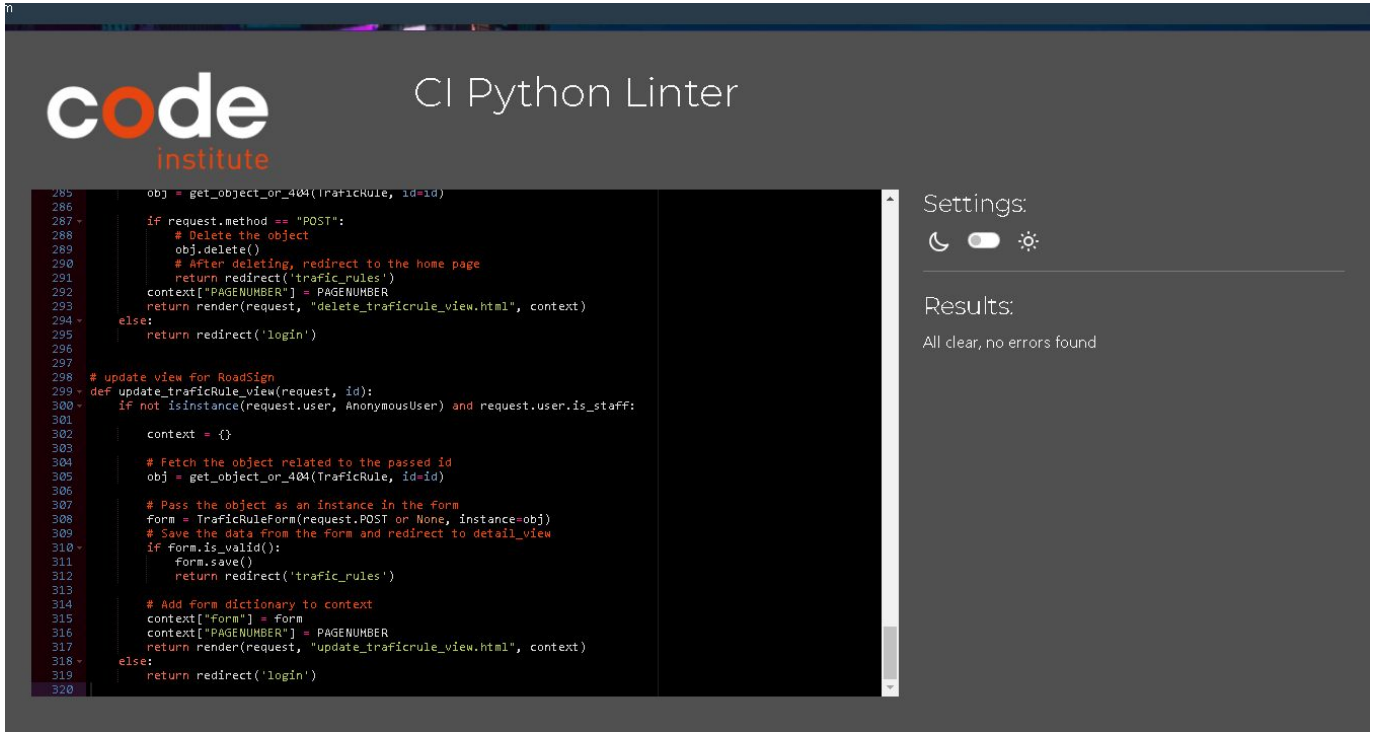
Views.py


code institute
CI Python Linter
obj = get_object_or_404(TraficRule, id=id)
if request.method == "POST":
# Delete the object
obj.delete()
# After deleting, redirect to the home page
return redirect('trafic_rules')
context["PAGENUMBER"] = PAGENUMBER
return render(request, "delete_traficrule_view.html", context)
else:
return redirect('login')
# update view for RoadSign
def update_traficRule_view(request, id):
if not isinstance(request.user, AnonymousUser) and request.user.is_staff:
context = {}
# Fetch the object related to the passed id
obj = get_object_or_404(TraficRule, id=id)
# Pass the object as an instance in the form
form = TraficRuleForm(request.POST or None, instance=obj)
# Save the data from the form and redirect to detail_view
if form.is_valid():
form.save()
return redirect('trafic_rules')
Add form dictionary to context
context["form"] = form
context["PAGENUMBER"] = PAGENUMBER
return render(request, "update_traficrule_view.html", context)
else:
return redirect('login')
Settings:
Results:
All clear, no errors found

models.py


code institute
CI Python Linter
def __str__(self):
return self.question.question_text
class TraficRule(models.Model):
title = models.CharField(max_length=400, validators=[
MinLengthValidator(5), MaxLengthValidator(400)])
sub_title = models.CharField(max_length=400, validators=[
MinLengthValidator(5),
MaxLengthValidator(400)])
sub_text = models.TextField(max_length=500, validators=[
MinLengthValidator(5),
MaxLengthValidator(500)])
image_link = models.CharField(max_length=500, validators=[
MinLengthValidator(5),
MaxLengthValidator(500)])
def __str__(self):
return self.title
class RoadSign(models.Model):
title = models.CharField(max_length=400, validators=[
MinLengthValidator(5),
MaxLengthValidator(400)], blank=False)
description = models.TextField(max_length=800, validators=[
MinLengthValidator(10),
MaxLengthValidator(800)], blank=False)
image_link = models.CharField(max_length=500, validators=[
MinLengthValidator(10),
MaxLengthValidator(500)],
blank=False)
def __str__(self):
return self.title
Settings:
Results:
All clear, no errors found

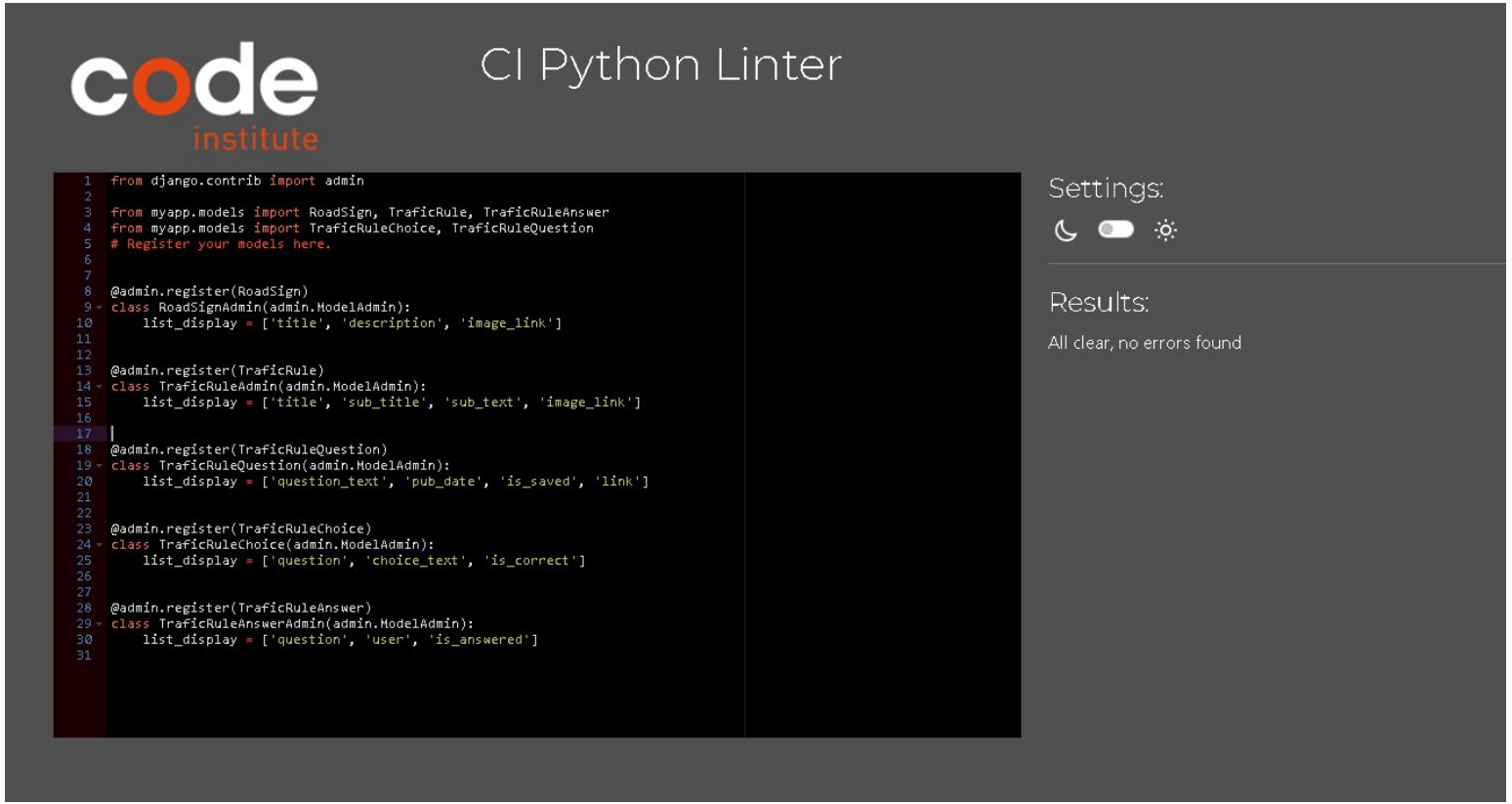
urls.py


code institute
CI Python Linter
from django.urls import path, include
from . import views
urlpatterns = [
path('trafic_rule_question_detail/<int:question_id>/',
views.trafic_rule_question_detail,
name='trafic_rule_question_detail'),
path("trafic_rules", views.trafic_rules, name="trafic_rules"),
path("", views.index, name="home"),
path('start_quiz', views.start_quiz, name='start_quiz'),
path('continue_quiz', views.continue_quiz, name='continue_quiz'),
path('next_question/<int:current_question_id>/',
views.next_question, name='next_question'),
path('previous_question/<int:current_question_id>/',
views.previous_question, name='previous_question'),
path('road_signs',
views.road_signs_page, name='road_signs'),
path('road_signs/create',
views.create_roadSign_view, name='road_signs/create'),
path('<int:id>/delete/',
views.delete_roadSign_view,
name='roadsign_delete'),
path('update/<int:id>/',
views.update_roadSign_view,
name='update_roadSign_view'),
path('trafic_rule/create',
views.create_traficrule_view, name='trafic_rule/create'),
path('delete/<int:id>/', views.delete_traficrule_view,
name='traficrule_delete'),
path('update/traficrule/<int:id>/', views.update_traficRule_view,
name='update_traficRule_view'),
]
Settings:
Results:
All clear, no errors found

forms.py


code institute
CI Python Linter
# specify fields to be used
fields = [
"title",
"description",
"image_link",
]
class TraficRuleForm(forms.ModelForm):
title = forms.CharField(widget=forms.TextInput(attrs={
'placeholder': 'Title',
'style': 'width: 300px;',
'class': 'form-control'}))
sub_title = forms.CharField(widget=forms.TextInput(attrs={
'placeholder': 'Subtitle',
'style': 'width: 300px;',
'class': 'form-control'}))
sub_text = forms.CharField(widget=forms.Textarea(attrs={
'placeholder': 'Subtext',
'style': 'width: 300px;',
'class': 'form-control'}))
image_link = forms.URLField(widget=forms.URLInput(attrs={
'placeholder': 'Image Link',
'style': 'width: 300px;',
'class': 'form-control'}))
class Meta:
model = TraficRule
fields = [
"title",
"sub_title",
"sub_text",
"image_link"
]
Settings:
Results:
All clear, no errors found

admin.py


code institute
CI Python Linter
from django.contrib import admin
from myapp.models import RoadSign, TraficRule, TraficRuleAnswer
from myapp.models import TraficRuleChoice, TraficRuleQuestion
# Register your models here.
@admin.register(RoadSign)
class RoadSignAdmin(admin.ModelAdmin):
list_display = ['title', 'description', 'image_link']
@admin.register(TraficRule)
class TraficRuleAdmin(admin.ModelAdmin):
list_display = ['title', 'sub_title', 'sub_text', 'image_link']
@admin.register(TraficRuleQuestion)
class TraficRuleQuestion(admin.ModelAdmin):
list_display = ['question_text', 'pub_date', 'is_saved', 'link']
@admin.register(TraficRuleChoice)
class TraficRuleChoice(admin.ModelAdmin):
list_display = ['question', 'choice_text', 'is_correct']
@admin.register(TraficRuleAnswer)
class TraficRuleAnswerAdmin(admin.ModelAdmin):
list_display = ['question', 'user', 'is_answered']
Settings:
Results:
All clear, no errors found

test_forms.py

code institute

CI Python Linter

```python
from django.test import SimpleTestCase
from myapp.forms import RoadSignForm, TraficRuleForm


class TestForms(SimpleTestCase):
    def test_roadsign_form_valid_data(self):
        form = RoadSignForm(data={
            "title": "roadsign 1",
            "description": "roadsign 1 description",
            "image_link": "firstimagelink.png"
        })

        self.assertTrue(form.is_valid())

    def test_roadsign_form_no_data(self):
        form = RoadSignForm(data={})

        self.assertFalse(form.is_valid())
        self.assertEqual(len(form.errors), 3)

    def test_traficrule_form_valid_data(self):
        form = TraficRuleForm(data={
            "title": "trafic rule 1",
            "sub_title": "This is the first trafic rule description",
            "sub_text": "This is the first trafic rule sub_text",
            "image_link": "firstimagelinkfortraficrule.png"
        })
        self.assertTrue(form.is_valid())

    def test_traficrule_form_no_data(self):
        form = RoadSignForm(data={})

        self.assertFalse(form.is_valid())
        self.assertEqual(len(form.errors), 3)

```

Settings:

Results:

All clear, no errors found

test_models.py

code institute

CI Python Linter

```python
    def test_road_sign_deletion(self):
        w = self.create_roadsign()
        initial_count = RoadSign.objects.count()
        w.delete()
        self.assertEqual(RoadSign.objects.count(), initial_count - 1)

    def create_traficrule(self, title="The first trafic rule title",
                          sub_title="The first trafic rule sub title",
                          sub_text="The first trafic rule sub text",
                          image_link="The first trafic rule image link"):
        return TraficRule.objects.create(title=title,
                                         sub_title=sub_title,
                                         sub_text=sub_text,
                                         image_link=image_link)

    def test_traficrule_creation(self):
        w = self.create_traficrule()
        self.assertEqual(w.title, "The first trafic rule title")

    def test_traffic_rule_deletion(self):
        w = self.create_traficrule()
        initial_count = TraficRule.objects.count()
        w.delete()
        self.assertEqual(TraficRule.objects.count(), initial_count - 1)

    def test_traffic_rule_update(self):
        w = self.create_traficrule()
        updated_title = "Updated traffic rule title"

        w.title = updated_title
        w.save()

        updated_traffic_rule = TraficRule.objects.get(pk=w.pk)

        self.assertEqual(updated_traffic_rule.title, updated_title)

```

Settings:

Results:

All clear, no errors found

test_urls.py

code institute

CI Python Linter

```python
    def test_trafic_rule_update_url_resolves(self):
        url = reverse("update_traficRule_view", args=[1])
        self.assertEqual(resolve(url).func, update_traficRule_view)

    def test_continue_quiz_url_resolves(self):
        url = reverse("continue_quiz")
        self.assertEqual(resolve(url).func, continue_quiz)

    def test_road_signs_page_url_resolves(self):
        url = reverse("road_signs")
        self.assertEqual(resolve(url).func, road_signs_page)

    def test_road_signs_create_url_resolves(self):
        url = reverse("road_signs/create")
        self.assertEqual(resolve(url).func, create_roadSign_view)

    def test_trafic_rule_question_detail_url_resolves(self):
        url = reverse("trafic_rule_question_detail", args=[1])
        self.assertEqual(resolve(url).func, trafic_rule_question_detail)

    def test_next_question_url_resolves(self):
        url = reverse("next_question", args=[1])
        self.assertEqual(resolve(url).func, next_question)

    def test_previous_question_url_resolves(self):
        url = reverse("previous_question", args=[1])
        self.assertEqual(resolve(url).func, previous_question)

    def test_delete_road_sign_url_resolves(self):
        url = reverse("roadsign_delete", args=[1])
        self.assertEqual(resolve(url).func, delete_roadSign_view)

    def test_update_road_sign_url_resolves(self):
        url = reverse("update_roadSign_view", args=[1])
        self.assertEqual(resolve(url).func, update_roadSign_view)

```

Settings:

Results:

All clear, no errors found

test_views.py

code institute

CI Python Linter

```
        }
        self.setup_user()

    def setup_user(self):
        User.objects.create_user(**self.credentials)

    def test_trafic_rules_view_deny_anonymous(self):
        response = self.client.get('/trafic_rules', follow=True)
        self.assertRedirects(response, '/login/')

    def test_trafic_rules_view_load(self):
        # Log in
        response_login = self.client.post(
            '/login/', self.credentials, follow=True)

        self.assertTrue(response_login.context['user'].is_active)

        response_traffic_rules = self.client.get('/trafic_rules', follow=True)

        self.assertEqual(response_traffic_rules.status_code, 200)

    def test_road_signs_page_view_deny_anonymous(self):
        response = self.client.get('/road_signs', follow=True)
        self.assertRedirects(response, '/login/')

    def test_road_signs_view_load(self):
        # Log in
        response_login = self.client.post(
            '/login/', self.credentials, follow=True)

        self.assertTrue(response_login.context['user'].is_active)

        response_road_signs = self.client.get('/road_signs', follow=True)

        self.assertEqual(response_road_signs.status_code, 200)

```

Settings:

Results:

All clear, no errors found