

## EVENTS

 CI Python Linter

apps.py


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

Settings:

 ☒ 

Results:



All clear, no errors found

 CI Python Linter

admin.py

```
1 from django.contrib import admin
2 from .models import Event, Category, EventAttendees
3
4 # Register your models here.
5 admin.site.register(Event)
6 admin.site.register(Category)
7 admin.site.register(EventAttendees)
8
```

Settings:

 ☒ 

Results:

All clear, no errors found

```
18
19
20 class Meta:
21     model = Event
22     fields = (
23         'event_title', 'description', 'category', 'event_image', 'date',
24         'start_time', 'end_time', 'location', 'limit',
25     )
26
27 class SearchForm(forms.ModelForm):
28
29     CHOICES = (
30         (1, 'Today'),
31         (2, 'Tomorrow'),
32         (3, 'This Weekend'),
33         (4, 'Next Week'),
34     )
35
36     when = forms.ChoiceField(choices=CHOICES)
37     date = forms.DateField(widget=forms.DateInput(
38         attrs={'type': 'date'}),
39         label='Pick a date')
40
41     location = forms.CharField()
42
43 class Meta:
44     model = Event
45     fields = (
46         'when', 'date', 'location', 'category',
47     )
48
```

Settings:



Results:

All clear, no errors found

```
98         raise ValidationError(errors)
99
100 def __str__(self):
101     return f"{self.event_title} | created by {self.host}"
102
103
104 class EventAttendees(models.Model):
105     """
106     Defines the EventAttendees model fields
107
108     :param models.Model: Django class to define data models
109     """
110     event = models.ForeignKey(
111         Event, on_delete=models.CASCADE, related_name="attending"
112     )
113     attendee = models.ForeignKey(
114         User, on_delete=models.CASCADE, related_name="attendee"
115     )
116     rsvp = models.IntegerField(choices=RESPONSE, default=0)
117     is_blocked = models.BooleanField(default=False)
118     created_at = models.DateTimeField(auto_now_add=True)
119
120 class Meta:
121     """
122     Specify the order of Attendees
123     """
124     ordering = ["-created_at"]
125
126 def __str__(self):
127     return f"{self.attendee} is going to {self.event}"
128
```

Settings:



Results:

All clear, no errors found



## CI Python Linter

test.py

```
1 from django.test import TestCase
2
3 # Create your tests here.
4
```

Settings:



Results:

All clear, no errors found



## CI Python Linter

url.py

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('<int:event_id>/', views.event_detail, name='event_detail'),
6     path('attending/', views.event_attending, name='event_attending'),
7     path('hosting/', views.event_hosting, name='event_hosting'),
8     path('create/', views.event_create, name='event_create'),
9     path('', views.EventList.as_view(), name='event_search'),
10    path('<int:event_id>/edit_event/',
11         views.event_edit, name='event_edit'),
12    path('<int:event_id>/event_delete/',
13         views.event_delete, name='event_delete'),
14    path('<int:event_id>/attend_event/',
15         views.attend_event, name='attend_event'),
16    path('<int:event_id>/remove_attendance/',
17         views.remove_attend_event, name='remove_attendance'),
18 ]
19
```

Settings:



Results:

All clear, no errors found



## CI Python Linter

views.py

```
187
188 is_events_none = future_events.count() == 0
189 is_past_events_none = past_events.count() == 0
190
191 paginator = Paginator(future_events, 9)
192 page_number = request.GET.get("page")
193 page_obj = paginator.get_page(page_number)
194
195 return render(
196     request,
197     "event/event_attending.html",
198     {
199         "page_obj": page_obj,
200         "past_events": past_events,
201         "is_events_none": is_events_none,
202         "is_past_events_none": is_past_events_none,
203     },
204 )
205
206
207 def attend_event(request, event_id):
208     event = Event.objects.get(id=event_id)
209     event.add_user_to_event(user=request.user)
210     return HttpResponseRedirect(reverse('event_attending'))
211
212
213 def remove_attend_event(request, event_id):
214     event = EventAttendees.objects.get(event=event_id, attendee=request.user)
215     event.delete()
216     return HttpResponseRedirect(reverse('event_attending'))
217
```

Settings:



Results:

All clear, no errors found

## USERS



## CI Python Linter

admin.py

```
23 fieldsets = (
24     (None, {"fields": (
25         "email",
26         "first_name",
27         "last_name",
28         "password",
29         "profile_image")
30     })),
31     ("Permissions", {"fields": ("is_active", "is_staff", "is_superuser")}),
32     ("Personal", {"fields": ("about", "location", "phone_number")})
33 )
34
35 add_fieldsets = (
36     (None, {
37         "classes": ("wide",),
38         "fields": (
39             "email",
40             "first_name",
41             "last_name",
42             "profile_image",
43             "password1",
44             "password2",
45             "is_active",
46             "is_staff"
47         )
48     })),
49 )
50
51 admin.site.register(NewUser, UserAdminConfig)
52
53
```

Settings:



Results:

All clear, no errors found



## CI Python Linter

apps.py

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

Settings:



Results:

All clear, no errors found



## CI Python Linter

forms.py

```
2 from cloudfinarys.forms import CloudinaryFileField
3 from phonenumber_field.formfields import PhoneNumberField
4 from .models import NewUser
5
6
7 class ProfileForm(forms.ModelForm):
8
9     TEL_MESSAGE = 'Incorrect format. Use an Irish number. e.g. (087) 456 7890.'
10    TEL_PLACEHOLDER = 'Irish Phone Numbers Only'
11
12    about = forms.CharField(widget=forms.Textarea(attrs={'rows': '3'}))
13    phone_number = PhoneNumberField(
14        widget=forms.TextInput(attrs={'placeholder': TEL_PLACEHOLDER}),
15        required=False,
16        region="IE"
17    )
18    phone_number.error_messages['invalid'] = TEL_MESSAGE
19    profile_image = CloudinaryFileField()
20
21    class Meta:
22        model = NewUser
23        fields = (
24            'first_name',
25            'last_name',
26            'email',
27            'profile_image',
28            'location',
29            'phone_number',
30            'about'
31        )
32
```

Settings:



Results:

All clear, no errors found



## CI Python Linter

models.py

```
69         "Superuser must be assigned to is_superuser = True"
70     )
71     return self.create_user(email, password, **other_fields)
72
73
74 class NewUser(AbstractBaseUser, PermissionsMixin):
75     """
76     Custom Django user model class for a new user.
77
78     :param AbstractBaseUser: Django class for creating custom user model
79     :param PermissionsMixin: Django mixin for user auth and permissions.
80     """
81
82     email = models.EmailField(unique=True)
83     first_name = models.CharField(max_length=150)
84     last_name = models.CharField(max_length=150)
85     profile_image = CloudinaryField("Profile Image", default="profile_image")
86     about = models.TextField(max_length=500, blank=True)
87     location = models.CharField(max_length=100, blank=True)
88     created_at = models.DateTimeField(auto_now=True)
89     phone_number = PhoneNumberField(blank=True)
90     is_active = models.BooleanField(default=True)
91     is_staff = models.BooleanField(default=False)
92
93     objects = CustomAccountManager()
94
95     USERNAME_FIELD = 'email'
96
97     def __str__(self):
98         return str(self.email)
99
```

Settings:



Results:

All clear, no errors found



## CI Python Linter

tests.py

```
1 from django.test import TestCase
2
3 # Create your tests here.
4
```

Settings:



Results:

All clear, no errors found



## CI Python Linter

urls.py

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

Settings:



Results:

All clear, no errors found



## CI Python Linter

views.py

```
14 - if request.method == "POST":
15     profile_form = ProfileForm(
16         data=request.POST,
17         files=request.FILES,
18         instance=user_profile
19     )
20     if profile_form.is_valid():
21         profile_form.save()
22         messages.add_message(
23             request,
24             messages.SUCCESS,
25             'Profile Successfully Updated'
26         )
27         return HttpResponseRedirect(reverse('user_profile'))
28     return render(
29         request,
30         "users/profile.html",
31         {
32             "profile_form": profile_form,
33         },
34     )
35
36 profile_form = ProfileForm(instance=user_profile)
37 return render(
38     request,
39     "users/profile.html",
40     {
41         "profile_form": profile_form,
42     },
43 )
44
```

Settings:



Results:

All clear, no errors found

### Whatson project app



## CI Python Linter

asgi.py

```
1 """
2 ASGI config for whatson project.
3
4 It exposes the ASGI callable as a module-level variable named ``application``.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/
8 """
9
10 import os
11
12 from django.core.asgi import get_asgi_application
13
14 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'whatson.settings')
15
16 application = get_asgi_application()
17 |
```

Settings:



Results:

All clear, no errors found

**Some of the standard Django settings are too long and cannot be shortened**



## CI Python Linter

settings.py

```
98 }
99
100 CSRF_TRUSTED_ORIGINS = [
101     "https://*.herokuapp.com"
102 ]
103
104 # Password validation
105 # https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators
106
107 AUTH_PASSWORD_VALIDATORS = [
108     {
109         'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
110     },
111     {
112         'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
113     },
114     {
115         'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
116     },
117     {
118         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
119     },
120 ]
121
122 # https://docs.allauth.org/en/dev/account/configuration.html
123 # No email verification user before signing in
124 ACCOUNT_EMAIL_VERIFICATION = "none"
125 # Set so no username is required when signing up
126 ACCOUNT_USER_MODEL_USERNAME_FIELD = None
127 # Email is required when signing up
```

Settings:



Results:

110: E501 line too long (91 > 79 characters)  
113: E501 line too long (81 > 79 characters)  
116: E501 line too long (82 > 79 characters)  
119: E501 line too long (83 > 79 characters)





## CI Python Linter

urls.py

```
1  """
2  URL configuration for whatson project.
3
4  The 'urlpatterns' list routes URLs to views. For more information please see:
5  https://docs.djangoproject.com/en/5.1/topics/http/urls/
6  Examples:
7  Function views
8      1. Add an import: from my_app import views
9      2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 from django.views.generic import TemplateView
20
21 urlpatterns = [
22     path('event/', include("event.urls"), name="event_urls"),
23     path('profile/', include("users.urls"), name="users_urls"),
24     path('accounts/', include("allauth.urls")),
25     path('admin/', admin.site.urls),
26     path('', TemplateView.as_view(template_name='home.html'), name="home")
27 ]
28
```

Settings:



Results:

All clear, no errors found



## CI Python Linter

wsgi.py

```
1  """
2  WSGI config for whatson project.
3
4  It exposes the WSGI callable as a module-level variable named ``application``.
5
6  For more information on this file, see
7  https://docs.djangoproject.com/en/5.1/howto/deployment/wsgi/
8  """
9
10 import os
11
12 from django.core.wsgi import get_wsgi_application
13
14 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'whatson.settings')
15
16 application = get_wsgi_application()
17
```

Settings:



Results:

All clear, no errors found