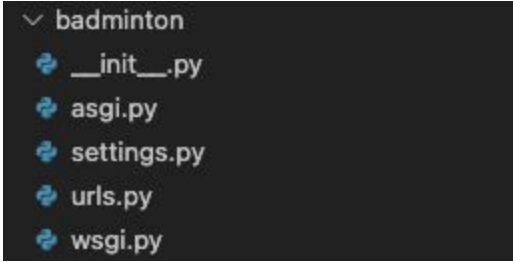


Project: Badminton



✓ badminton

- 🔗 `__init__.py`
- 🔗 `asgi.py`
- 🔗 `settings.py`
- 🔗 `urls.py`
- 🔗 `wsgi.py`

This image shows a file explorer window with a dark background. At the top, there is a folder icon and the text 'badminton'. Below it, there are five files, each preceded by a blue icon that looks like a small gear or a document with a plus sign. The files are listed as follows: '__init__.py', 'asgi.py', 'settings.py', 'urls.py', and 'wsgi.py'.

asgi.py



CI Python Linter

```
1 import os
2
3 from django.core.asgi import get_asgi_application
4
5 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'badminton.settings')
6
7 application = get_asgi_application()
8
```

Settings:



Results:

All clear, no errors found

settings.py



CI Python Linter

```
1 """
2 Django settings for badminton project.
3
4 Generated by 'django-admin startproject' using Django 4.2.11.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/4.2/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/4.2/ref/settings/
11 """
12
13 from pathlib import Path
14 import os
15 import dj_database_url
16 if os.path.isfile('env.py'):
17     import env
```

Settings:



Results:

[121](#): E501 line too long (91 > 79 characters)
[124](#): E501 line too long (81 > 79 characters)
[127](#): E501 line too long (82 > 79 characters)
[130](#): E501 line too long (83 > 79 characters)

Some lines of standard Django code are too long and impossible to reduce.

urls.py



CI Python Linter

```
1 from django.contrib import admin
2 from django.urls import path, include
3
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('events/', include('events.urls')),
7     path('accounts/', include('allauth.urls')),
8     path('accounts/profile/', include('profiles.urls')),
9     path('', include('home.urls')),
10 ]
11
```

Settings:



Results:

All clear, no errors found

wsgi.py



CI Python Linter

```
1 import os
2
3 from django.core.wsgi import get_wsgi_application
4
5 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'badminton.settings')
6
7 application = get_wsgi_application()
8
```

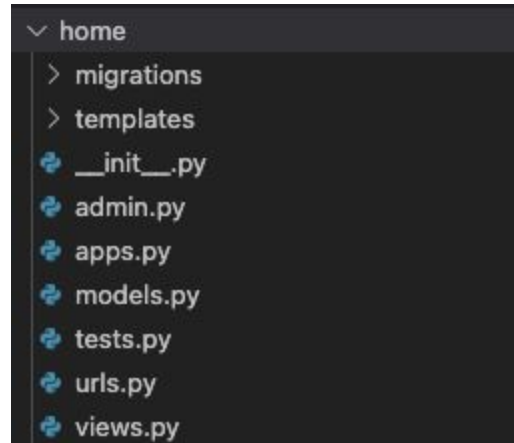
Settings:



Results:

All clear, no errors found

App: Home



apps.py



CI Python Linter

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

Settings:



Results:

All clear, no errors found

urls.py



CI Python Linter

```
1 """urls for the home app"""
2 from django.urls import path
3 from . import views
4
5 urlpatterns = [
6     path('', views.Home.as_view(), name='home'),
7 ]
8
```

Settings:



Results:

All clear, no errors found

views.py



CI Python Linter

```
1  """view for the home game"""
2  from django.views.generic import TemplateView
3
4
5  class Home(TemplateView):
6      template_name = 'home/index.html'
7  |
```

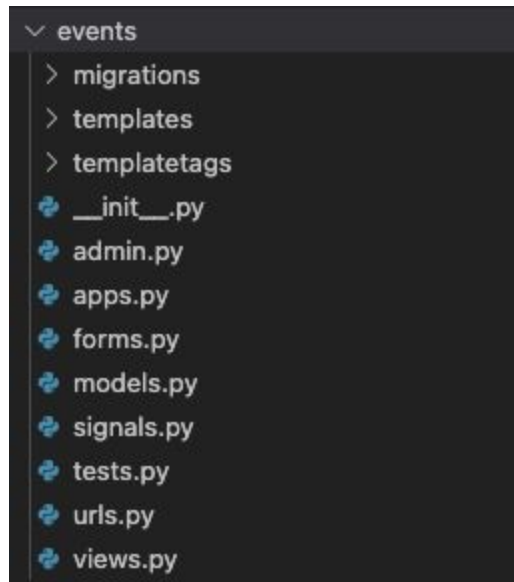
Settings:



Results:

All clear, no errors found

App: Events



admin.py



CI Python Linter

```
1 '''Admin for the Events app'''
2 from django.contrib import admin
3 from .models import *
4
5
6 admin.site.register(Group)
7 admin.site.register(Session)
8 admin.site.register(Roster)
9 admin.site.register(Team)
10 admin.site.register(Fixture)
11 admin.site.register(Game)
12 admin.site.register(Matchup)
13
```

Settings:



Results:

All clear, no errors found

apps.py



CI Python Linter

```
1 '''App for the Events app'''
2 from django.apps import AppConfig
3
4
5 class EventsConfig(AppConfig):
6     default_auto_field = 'django.db.models.BigAutoField'
7     name = 'events'
8
9     def ready(self):
10         import events.signals
11 |
```

Settings:



Results:

All clear, no errors found

forms.py



CI Python Linter

```
1  '''Forms for the Events app'''
2  from django import forms
3  from django.db.models import Q
4  from django.forms import Form, ModelForm, HiddenInput
5  from .models import Game, Group, Session, Matchup
6  from django.contrib.auth.models import User
7  from profiles.models import SessionInvite, Profile
8
9
10 class GroupCreationForm(ModelForm):
11     '''Form to create new group'''
12     def __init__(self, *args, **kwargs):
13         '''Add bootstrap floating input field styling'''
14         super(GroupCreationForm, self).__init__(*args, **kwargs)
15         for field in self.fields:
16             if field != 'private':
17                 form_id = 'floating' + field.capitalize()
```

Settings:



Results:

All clear, no errors found

models.py



CI Python Linter

```
1  '''Models for the events app'''
2  from django.db import models
3  from django.contrib.auth.models import User
4  from django.db.models import Q, Count, Max
5
6
7  class Group(models.Model):
8      """Group event model"""
9      name = models.CharField(max_length=30, unique=True, blank=False)
10     host = models.ForeignKey(
11         User, on_delete=models.DO_NOTHING, related_name='group_author')
12     created_at = models.DateTimeField(auto_now_add=True)
13     updated_at = models.DateTimeField(auto_now=True)
14     private = models.BooleanField(default=False)
15
16     def __str__(self):
17         """group model string representation"""
```

Settings:



Results:

All clear, no errors found

signals.py



CI Python Linter

```
1 '''Signals for the event app'''
2 from django.db.models.signals import post_save, pre_save
3 from django.dispatch import receiver
4 from .models import Game, Team, Matchup, Session
5
6
7 @receiver(pre_save, sender=Session)
8 def pre_save_session(sender, instance, **kwargs):
9     '''signal to change joinable status for session'''
10     print(instance.status)
11     if instance.status == 1:
12         instance.joinable = True
13     else:
14         instance.joinable = False
15
16
17 @receiver(post_save, sender=Game)
```

Settings:



Results:

All clear, no errors found

urls.py



CI Python Linter

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('groups/', views.GroupListView.as_view(), name="groups"),
6     path(
7         'groups/create/',
8         views.CreateGroupView.as_view(),
9         name="group_create"
10    ),
11    path('groups/<pk>/', views.GroupDetailView.as_view(), name="group_detail"),
12    path(
13        'groups/<pk>/update/',
14        views.UpdateGroupView.as_view(),
15        name="group_update"
16    ),
17    path(
```

Settings:



Results:

All clear, no errors found

views.py



CI Python Linter

```
1  '''Views for Events app'''
2  from django.views.generic.edit import CreateView, UpdateView
3  from django.views.generic.edit import FormMixin, DeleteView
4  from django.contrib.auth.mixins import AccessMixin
5  from django.views.generic import DetailView, ListView
6  from django.contrib.auth.mixins import LoginRequiredMixin
7  from django.shortcuts import get_object_or_404, redirect
8  from django.urls import reverse_lazy
9  from .forms import *
10 from .models import *
11
12
13 class GroupListView(ListView):
14     '''Group list view'''
15     model = Group
16     template_name = 'events/group_list.html'
17     context_object_name = 'list_object'
```

Settings:



Results:

All clear, no errors found

model_method.py



CI Python Linter

```
1 '''template tags for passing variables in method'''
2 from django import template
3 register = template.Library()
4
5
6 @register.filter(name='grp_session_wins')
7 def call_method(obj, grp):
8     return obj.profile.get_session_wins(grp)
9
10
11 @register.filter(name='grp_game_wins')
12 def call_method(obj, grp):
13     return obj.profile.get_group_game_wins(grp)
14
15
16 @register.filter(name='profile_pairing')
17 def call_method(obj, profile):
```

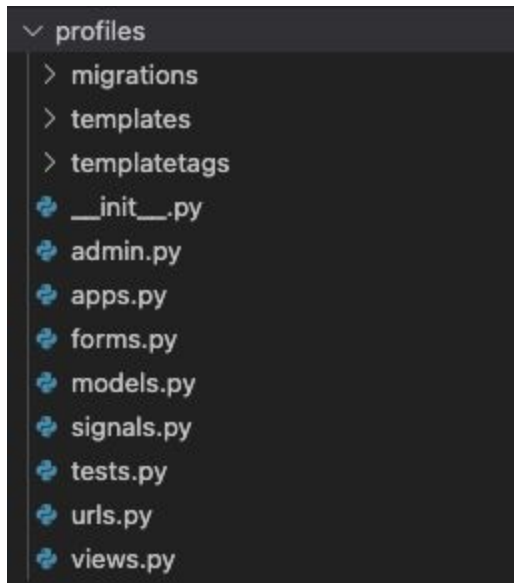
Settings:



Results:

All clear, no errors found

App: Profile



admin.py



CI Python Linter

```
1 '''Admin for profile app'''
2 from django.contrib import admin
3 from .models import Profile, FriendRequest, SessionInvite
4
5
6 admin.site.register(Profile)
7 admin.site.register(FriendRequest)
8 admin.site.register(SessionInvite)
9
```

Settings:



Results:

All clear, no errors found

apps.py



CI Python Linter

```
1 '''Apps for profile app'''
2 from django.apps import AppConfig
3
4
5 class ProfilesConfig(AppConfig):
6     default_auto_field = 'django.db.models.BigAutoField'
7     name = 'profiles'
8
9     def ready(self):
10         import profiles.signals
11
```

Settings:



Results:

All clear, no errors found

forms.py



CI Python Linter

```
1  '''Forms for profile app'''
2  from django.forms import Form, ModelForm
3  from allauth.account.forms import LoginForm, SignupForm
4  from django.contrib.auth.models import User
5  from .models import *
6
7
8  class MyCustomLoginForm(LoginForm):
9      '''Login Form'''
10     def __init__(self, *args, **kwargs):
11         '''Add bootstrap floating input field styling'''
12         super(MyCustomLoginForm, self).__init__(*args, **kwargs)
13         for field in self.fields:
14             if field != 'remember':
15                 form_id = 'floating' + field.capitalize()
16                 self.fields[field].widget.attrs.update(
17                     {
```

Settings:



Results:

All clear, no errors found

models.py



CI Python Linter

```
1 '''models for profile app'''
2 from django.db import models
3 from django.contrib.auth.models import User
4 from django.db.models import Q
5 from events.models import *
6
7
8 class Profile(models.Model):
9     """Profile model"""
10     user = models.OneToOneField(
11         User,
12         on_delete=models.CASCADE,
13         related_name='profile'
14     )
15     first_name = models.CharField(max_length=25, blank=True, null=True)
16     last_name = models.CharField(max_length=25, blank=True, null=True)
17     friends = models.ManyToManyField('Profile', blank=True)
```

Settings:



Results:

All clear, no errors found

signals.py



CI Python Linter

```
5 from profiles.models import Profile, FriendRequest, SessionInvite
6
7
8 @receiver(post_save, sender=User)
9 def create_profile(sender, instance, created, **kwargs):
10     """Signal to create profile for a user"""
11     if created:
12         Profile.objects.create(user=instance)
13
14
15 @receiver(post_save, sender=FriendRequest)
16 def post_save_add_friend(sender, created, instance, **kwargs):
17     '''Signal add freind to both users and delete friend request'''
18     sender_ = instance.sender
19     receiver_ = instance.receiver
20     if instance.status == 'accepted':
```

Settings:



Results:

All clear, no errors found

urls.py



CI Python Linter

```
1 '''urls for profile app'''
2 from django.urls import path
3 from . import views
4
5 urlpatterns = [
6     path('', views.MyProfileView.as_view(), name="profile"),
7     path('update/', views.UpdateProfile.as_view(), name="update_profile"),
8     path('search/', views.FriendSearchView.as_view(), name="search_profile"),
9     path('<pk>', views.UserProfileView.as_view(), name="user_profile"),
10    path(
11        '<pk>/remove/',
12        views.RemoveFriendView.as_view(),
13        name="remove_profile"
14    ),
15    path(
16        'request/search/',
17        views.FriendRequestListView.as_view(),
```

Settings:



Results:

All clear, no errors found

views.py



CI Python Linter

```
1 '''Views for profile app'''
2 from django.contrib.auth.mixins import AccessMixin
3 from django.views.generic.edit import UpdateView, FormMixin, DeleteView
4 from django.views.generic import DetailView, ListView, TemplateView
5 from django.contrib.auth.mixins import LoginRequiredMixin
6 from django.shortcuts import redirect, get_object_or_404
7 from django.urls import reverse_lazy
8 from .forms import *
9 from .models import *
10
11
12 class MyProfileView(LoginRequiredMixin, DetailView):
13     '''Profile view for current user'''
14     model = Profile
15     template_name = 'profiles/profile.html'
16     context_object_name = 'user_object'
17
```

Settings:



Results:

All clear, no errors found

add_attr.py



CI Python Linter

```
1 '''filter for bootstrap form validation css'''
2 from django import template
3 register = template.Library()
4
5
6 @register.filter(name='add_attr')
7 def add_attribute(field, css):
8     # Accessing the already declared attributes
9     attrs = field.subwidgets[0].data['attrs']
10    definition = css.split(',')
11
12    for d in definition:
13        if ':' not in d:
14            attrs['class'] += f" {d}" # Extending the class string
15        else:
16            key, val = d.split(':')
17            attrs[key] += f'{val}' # Extending the `key` string
```

Settings:



Results:

All clear, no errors found