```
"""
ASGI config for JobPortal project.

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

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

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

application = get_asgi_application()

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'aralxt83cxseu%+7%(-wx3qrtf+xjrq64zg9lxw&88coqr3ha*'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'jobs',
]
```

```python
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'JobPortal.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ["jobs/templates"],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'JobPortal.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-
validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidat
or',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "jobs/static")
]

MEDIA_ROOT = os.path.join(BASE_DIR, 'jobs/media')
MEDIA_URL = '/media/'
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('jobs.urls')),

]  + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

"""
WSGI config for JobPortal 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/3.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

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

application = get_wsgi_application()

"""Activate virtualenv for current interpreter:

Use exec(open(this_file).read(), {'__file__': this_file}).

This can be used when you must use an existing Python interpreter, not
the virtualenv bin/python.
"""
import os
import site
import sys

try:
    abs_file = os.path.abspath(__file__)
except NameError:
    raise AssertionError("You must use exec(open(this_file).read(),
{'__file__': this_file}))")

bin_dir = os.path.dirname(abs_file)
base = bin_dir[: -len("Scripts") - 1]  # strip away the bin part from
the __file__, plus the path separator

# prepend bin to PATH (this file is inside the bin directory)
os.environ["PATH"] = os.pathsep.join([bin_dir] + os.environ.get("PATH",
"").split(os.pathsep))
```

```
os.environ["VIRTUAL_ENV"] = base  # virtual env is right above bin
directory

# add the virtual environments libraries to the host python import
mechanism
prev_length = len(sys.path)
for lib in "..\Lib\site-packages".split(os.pathsep):
    path = os.path.realpath(os.path.join(bin_dir, lib))
    site.addsitedir(path.decode("utf-8") if "" else path)
sys.path[:] = sys.path[prev_length:] + sys.path[0:prev_length]

sys.real_prefix = sys.prefix
sys.prefix = base
```