HR Recruitment — Django + DRF Scaffold

This document contains a ready-to-copy **Django project scaffold** for your HR recruitment app (public job apply link + ATS pipeline). Paste the files into your project folder and follow the setup steps in the **README** section below.

What is included

- `requirements.txt`
- `README.md` (setup + run commands)
- Django project: `hr_recruitment/` (settings, urls, wsgi)
- Django app: `recruitment/` (models, serializers, views, urls, admin)
- Example `manage.py`

Project structure (what to create)

```
hr_recruitment_project/
├─ manage.py
├─ requirements.txt
├─ README.md
├─ hr_recruitment/
│  ├─ __init__.py
│  ├─ settings.py
│  ├─ urls.py
│  └─ wsgi.py
└─ recruitment/
   ├─ __init__.py
   ├─ admin.py
   ├─ apps.py
   ├─ models.py
   ├─ serializers.py
   ├─ views.py
   ├─ urls.py
   └─ migrations/
      └─ __init__.py
```

---

requirements.txt

```
Django>=4.2
djangorestframework
django-cors-headers
python-dotenv
gunicorn
```

---

manage.py

```python
#!/usr/bin/env python
import os
import sys

if __name__ == '__main__':
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'hr_recruitment.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and available
on your PYTHONPATH environment variable?"
        ) from exc
    execute_from_command_line(sys.argv)
```

---

hr_recruitment/settings.py

```python
import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = os.environ.get('DJANGO_SECRET_KEY', 'dev-secret-key-change-me')
DEBUG = os.environ.get('DJANGO_DEBUG', '1') == '1'
ALLOWED_HOSTS = os.environ.get('DJANGO_ALLOWED_HOSTS', '*').split(',') if not
DEBUG else []

# Installed apps
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Third party
    'rest_framework',
    'corsheaders',

    # Local
    'recruitment',
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    '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 = 'hr_recruitment.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        '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 = 'hr_recruitment.wsgi.application'

# Database: default sqlite for dev
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation (default)
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Asia/Karachi'
USE_I18N = True
USE_TZ = True

# Static & media
STATIC_URL = '/static/'
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

# DRF defaults (simple)
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
}

# CORS (dev) — allow all origins for now; restrict in production
CORS_ALLOW_ALL_ORIGINS = True

# Default primary key field type
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

hr_recruitment/urls.py

```
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('api/', include('recruitment.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

hr_recruitment/wsgi.py

```
import os
from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'hr_recruitment.settings')
application = get_wsgi_application()
```

recruitment/apps.py

```
from django.apps import AppConfig

class RecruitmentConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'recruitment'
```

recruitment/models.py

```
from django.db import models

class Job(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField(blank=True)
    location = models.CharField(max_length=100, blank=True)
    is_open = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

class CandidateApplication(models.Model):
    STAGES = [
        ('applied', 'Applied'),
        ('phone', 'Phone Screen'),
        ('interview', 'Interview'),
        ('rejected', 'Rejected'),
    ]
    job = models.ForeignKey(Job, related_name='applications',
on_delete=models.CASCADE)
    full_name = models.CharField(max_length=200)
    email = models.EmailField()
    resume = models.FileField(upload_to='resumes/')
    stage = models.CharField(max_length=20, choices=STAGES, default='applied')
    applied_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.full_name} - {self.job.title} ({self.stage})"
```

---

recruitment/serializers.py

```
from rest_framework import serializers
from .models import Job, CandidateApplication

class JobSerializer(serializers.ModelSerializer):
    class Meta:
        model = Job
        fields = ['id','title','description','location','is_open','created_at']

class CandidateApplicationSerializer(serializers.ModelSerializer):
    class Meta:
        model = CandidateApplication
        fields = ['id','job','full_name','email','resume','stage','applied_at']
        read_only_fields = ['stage','applied_at']
```

---

recruitment/views.py

```
from rest_framework import viewsets, permissions, status
from rest_framework.decorators import action
from rest_framework.response import Response
from .models import Job, CandidateApplication
from .serializers import JobSerializer, CandidateApplicationSerializer

class JobViewSet(viewsets.ModelViewSet):
    """Public job listing and admin CRUD."""
    queryset = Job.objects.all().order_by('-created_at')
    serializer_class = JobSerializer

    @action(detail=True, methods=['post'],
permission_classes=[permissions.AllowAny])
    def apply(self, request, pk=None):
        """Public endpoint: POST /api/jobs/{id}/apply/ with form-data
(full_name, email, resume)"""
        job = self.get_object()
        data = request.data.copy()
        data['job'] = job.id
        serializer = CandidateApplicationSerializer(data=data)
        if serializer.is_valid():
            serializer.save()
            # TODO: trigger email notification here (send confirmation to
candidate)
            return Response({'status': 'Application submitted'},
status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

class ApplicationViewSet(viewsets.ModelViewSet):
    """Protected: list and update application stage. Keep default perms
(IsAuthenticated)"""
    queryset =
CandidateApplication.objects.select_related('job').all().order_by('-applied_at')
    serializer_class = CandidateApplicationSerializer

    # Optionally allow patching only the 'stage' field for moving in pipeline
    def partial_update(self, request, *args, **kwargs):
        # Only allow stage updates from HR UI if desired
        return super().partial_update(request, *args, **kwargs)
```

recruitment/urls.py

```python
from rest_framework.routers import DefaultRouter
from .views import JobViewSet, ApplicationViewSet

router = DefaultRouter()
router.register(r'jobs', JobViewSet, basename='job')
router.register(r'applications', ApplicationViewSet, basename='application')

urlpatterns = router.urls
```

---

recruitment/admin.py

```python
from django.contrib import admin
from .models import Job, CandidateApplication

@admin.register(Job)
class JobAdmin(admin.ModelAdmin):
    list_display = ('title','location','is_open','created_at')
    list_filter = ('is_open',)
    search_fields = ('title','description')

@admin.register(CandidateApplication)
class CandidateApplicationAdmin(admin.ModelAdmin):
    list_display = ('full_name','email','job','stage','applied_at')
    list_filter = ('stage','job')
    search_fields = ('full_name','email')
```

README (quick setup & run)

```
# Quick setup (local)

# 1. create venv & install
python -m venv venv
source venv/bin/activate  # or `venv\Scripts\activate` on Windows
pip install -r requirements.txt

# 2. create project files (paste files from scaffold into folders)

# 3. makemigrations & migrate
```

```
python manage.py makemigrations
python manage.py migrate

# 4. create superuser (for HR admin dashboard)
python manage.py createsuperuser

# 5. run server
python manage.py runserver

# 6. Access
# - Admin: http://127.0.0.1:8000/admin/
# - API: http://127.0.0.1:8000/api/jobs/ (GET)
# - Apply (example): POST http://127.0.0.1:8000/api/jobs/1/apply/ (multipart/
form-data)

# Example curl (submit resume):
# Ensure file `resume.pdf` exists locally
curl -X POST -F "full_name=Jane Doe" -F "email=jane@example.com" -F
"resume=@resume.pdf" http://127.0.0.1:8000/api/jobs/1/apply/

# Notes:
# - In development DEBUG=True serves media files. In production, configure MEDIA
settings & secure uploads.
# - REST endpoints for applications are protected by default (IsAuthenticated).
Use Django admin to view applications, or build HR frontend.
# - To allow the HR dashboard (React) to call the API from another origin,
CORS_ALLOW_ALL_ORIGINS is set True (dev only).
```

Next immediate steps (suggested)

1. Create the project files locally from the scaffold above.
2. Run migrations and create superuser.
3. Create a job via admin or API.
4. Test application endpoint with curl or Postman.

When you're ready I can: - Generate a **Netlify-ready React public apply page** that hits `/api/jobs/{id}/apply/` (no login). OR - Generate the **HR React dashboard** with drag-and-drop pipeline that calls `/api/applications/{id}/` PATCH to update stage. - Add **email notifications** (console & Gmail SMTP) and an example Celery task.

Tell me which of those I should produce next and I'll generate the code + instructions.

End of scaffold.