1. Create a folder project in your workspace.
2. Create virtual environment – ***python -m venv .venv***

A screenshot of a computer

Description automatically generated

1. Activate virtual environment - ***.\.venv\Scripts\activate***

A screen shot of a computer code

Description automatically generated

1. Install Django framework in the virtual environment -

***pip install django***

A screenshot of a computer program

Description automatically generated

1. Create python project backend –

***django-admin startproject backend***

A screenshot of a computer

Description automatically generated

1. Create react project frontend -

***npx create-react-app frontend***

A screen shot of a computer

Description automatically generated

1. After creating frontend and backend

A screenshot of a computer screen

Description automatically generated

1. Create Django application volunteer inside backend

***cd backend***

***python manage.py startup volunteer***

A screenshot of a computer

Description automatically generated

1. Install Django Rest Framework (DRF) –

***pip install djangorestframework***

Powerful and flexible toolkit for building RESTful APIs in Django. It simplifies the process of creating APIs, providing features like serialization, authentication, and view handling.

A screen shot of a computer

Description automatically generated

1. Install cors headers -

***pip install Django-cors-headers***

A screen shot of a computer

Description automatically generated

Enable and configure **Cross-Origin Resource Sharing (CORS)** in Django applications. CORS is a security mechanism that restricts how resources on a web server can be accessed by scripts running in a browser on a different origin.

**Purpose of django-cors-headers**

* **Handle Cross-Origin Requests**: Allow frontend applications (e.g., React, Angular) hosted on different domains or ports to interact with a Django backend.
* **Ensure Secure API Access**: Configure which domains are allowed to make requests to the Django server.
* **Simplify Development**: Facilitate local development setups where the frontend and backend are typically served from different origins.

**Key Features**

* Allows you to specify **permitted origins** for cross-origin requests.
* Supports fine-grained configuration for methods, headers, and credentials.
* Can allow all origins or restrict to a specific list of trusted domains.
* Integrates seamlessly into the Django middleware stack.

1. Install MySQL Client

***pip install mysqlclient***

A screen shot of a computer program

Description automatically generated

1. Change the settings

Include the app, rest framework, and cors headers

***'rest\_framework',***

***'corsheaders',***

***'volunteer',***

A screenshot of a computer

Description automatically generated

1. Change the settings

Add cors middlewares - ***'corsheaders.middleware.CorsMiddleware',***

A computer screen with text

Description automatically generated

1. Change the settings - Configure Allowed Origins

***CORS\_ALLOWED\_ORIGINS = [***

***"http://localhost:3000",***

***"http://127.0.0.1:3000",***

***]***

***A black screen with numbers and text

Description automatically generated***

1. Add media settings to manage media resources such as images, video and audio files

***import os***

***# Media files (uploads)***

***MEDIA\_URL = '/media/'***

***MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')***

1. Change the database settings

***DATABASES = {***

***'default': {***

***'ENGINE' : 'django.db.backends.mysql',***

***'NAME' : 'hopehands\_demo', # your DB name***

***'USER' : 'root', # your DB user name***

***'PASSWORD': 'Admin', # your DB password***

***'HOST' : 'localhost', # your DB host***

***'PORT' : '3306', # your DB port***

***}***

***}***

A screenshot of a computer program

Description automatically generated

1. Define volunteer model – models.py

***from django.db import models***

***# Create your models here.***

***from django.db import models***

***# Create your models here.***

***class Profile(models.Model):***

***STATUS\_CHOICES = [***

***('active', 'Active'),***

***('inactive', 'Inactive'),***

***]***

***# Profile picture***

***profile\_picture = models.ImageField(upload\_to='profile\_pictures/', blank=True, null=True)***

***# Basic Information***

***first\_name = models.CharField(max\_length=50)***

***last\_name = models.CharField(max\_length=50)***

***email = models.EmailField(unique=True)***

***# Optional fields***

***phone\_number = models.CharField(max\_length=15, blank=True, null=True)***

***date\_of\_birth = models.DateField(blank=True, null=True)***

***address = models.TextField(blank=True, null=True)***

***# Registration Information***

***registration\_date = models.DateField()***

***availability = models.CharField(max\_length=100, blank=True, null=True)***

***# Status with predefined choices***

***status = models.CharField(max\_length=20, choices=STATUS\_CHOICES, default='active')***

***def \_\_str\_\_(self):***

***return f"{self.first\_name} {self.last\_name}"***

1. Add Serializer for volunteer - serializers.py

***# serializers.py***

***from rest\_framework import serializers***

***from .models import Profile***

***class ProfileSerializer(serializers.ModelSerializer):***

***class Meta:***

***model = Profile***

***fields = '\_\_all\_\_'***

***extra\_kwargs = {***

***'password': {'write\_only': True}***

***}***

1. Add api views using ViewsSet

***# api\_views.py***

***from rest\_framework import viewsets***

***# Importing Profile model from models.py - Change according to your model***

***from .models import Profile***

***# Importing ProfileSerializer from serializers.py - Change according to your serializer***

***from .serializers import ProfileSerializer***

***class ProfileViewSet(viewsets.ModelViewSet):***

***queryset = Profile.objects.all()***

***serializer\_class = ProfileSerializer***

1. Register urls with router

***# volunteer app - urls.py***

***from django.urls import path, include***

***from rest\_framework.routers import DefaultRouter***

***from .api\_views import ProfileViewSet***

***router = DefaultRouter()***

***router.register(r'profiles', ProfileViewSet)***

***urlpatterns = [***

***path('', include(router.urls)),***

***]***

1. Include application urls in project

***# backend/urls.py***

***from django.contrib import admin***

***from django.urls import path, include***

***urlpatterns = [***

***path('admin/', admin.site.urls),***

***path('api/v1/volunteer/', include('volunteer.urls')),***

***]***

1. Make Migrate the model to create database tables

***python manage.py makemigrations***

A black screen with red text

Description automatically generated

To fix this problem install pillow.

A screenshot of a computer program

Description automatically generated

1. Migrate the model to create database tables

***python manage.py migrate***

A computer screen shot of a program

Description automatically generated

1. DB structure after migration

A screenshot of a computer

Description automatically generated

1. Run the application

***python manage.py runserver***

A screen shot of a computer program

Description automatically generated

1. Test volunteer API

[***http://localhost:8000***](http://localhost:8000)

A screenshot of a computer

Description automatically generated

[***http://localhost:8000/api/v1/volunteer/profiles/***](http://localhost:8000/api/v1/volunteer/profiles/)

A screenshot of a computer

Description automatically generated

Fill the details and Post

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

To view a record - http://localhost:8000/api/v1/volunteer/profiles/1/

A screenshot of a computer

Description automatically generated

Make changes and put to edit the record

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now add Authentication using JWT Token

1. Install JWT Token library

***pip install PyJWT***

***pip install djangorestframework-simplejwt***

1. Change the settings

***from datetime import timedelta***

***REST\_FRAMEWORK = {***

***"DEFAULT\_AUTHENTICATION\_CLASSES": (***

***"rest\_framework\_simplejwt.authentication.JWTAuthentication",***

***),***

***"DEFAULT\_PERMISSION\_CLASSES": [***

***"rest\_framework.permissions.IsAuthenticated",***

***],***

***}***

***SIMPLE\_JWT = {***

***"ACCESS\_TOKEN\_LIFETIME": timedelta(minutes=30),***

***"REFRESH\_TOKEN\_LIFETIME": timedelta(days=1),***

***}***

***CORS\_ALLOWS\_CREDENTIALS = True***

1. Test the URL after changing the settings

***A screenshot of a video

Description automatically generated***

1. Include api auth urls to generate JWT token and access data

path("api-auth/", include("rest\_framework.urls")),

***# backend/urls.py***

***from django.contrib import admin***

***from django.urls import path, include***

***urlpatterns = [***

***path('admin/', admin.site.urls),***

***path('api/v1/volunteer/', include('volunteer.urls')),***

***path("api-auth/", include("rest\_framework.urls")),***

***]***

1. Generate the token and test

A screenshot of a computer

Description automatically generated

1. Profile model doesn’t have the password and user profile is not created to authenticate. Revise the model to make the necessary changes
   1. Add password field
   2. Use the basic information to create authentication user model
   3. Modify models.py, serializers.py, api\_views.py
2. Test API

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a video chat

Description automatically generated

1. Exclude authentication for creating a profile by introducing custom permissions

* Create permissions.py

***#volunteer/permissions.py***

***from rest\_framework import permissions***

***class IsAuthenticatedOrCreate(permissions.BasePermission):***

***"""***

***Custom permission to allow access to profile creation without authentication.***

***For other actions, authentication is required.***

***"""***

***def has\_permission(self, request, view):***

***# Allow access for the 'POST' method to create a profile (no authentication needed)***

***if request.method == 'POST':***

***return True***

***# For other methods (GET, PUT, DELETE), check if the user is authenticated***

***return request.user and request.user.is\_authenticated***

* Alter api\_views.py

***# volunteer/api\_views.py***

*from rest\_framework import generics*

*from .models import Profile*

*from .serializers import ProfileSerializer*

***from .permissions import IsAuthenticatedOrCreate***

*# View to handle listing and creating profiles*

*class ProfileListCreateView(generics.ListCreateAPIView):*

*queryset = Profile.objects.all()  # Fetch all profile objects from the database*

*serializer\_class = ProfileSerializer  # Use the ProfileSerializer for validation and data handling*

***permission\_classes = [IsAuthenticatedOrCreate]  # Apply the custom permission class***

*# View to handle retrieving, updating, and deleting a single profile*

*class ProfileDetailView(generics.RetrieveUpdateDestroyAPIView):*

*queryset = Profile.objects.all()  # Fetch all profile objects from the database*

*serializer\_class = ProfileSerializer  # Use the ProfileSerializer for validation and data handling*

1. Test API

A screenshot of a computer

Description automatically generated

Prior to post check the DB tables

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Migrate the model changes and check

After Migration

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Fill up the form and post

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

View Profile Details require authentication

A screenshot of a computer

Description automatically generated

Therefore, we need to login before view the details

A screenshot of a computer

Description automatically generated

To generate tokens

from rest\_framework\_simplejwt.views import TokenObtainPairView, TokenRefreshView

     # Route for obtaining a JWT token

    path("api/token/", TokenObtainPairView.as\_view(), name="get\_token"),

    # Route for refreshing an expired JWT token

    path("api/token/refresh/", TokenRefreshView.as\_view(), name="refresh"),

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Generate JWT token using the credentials username and password

A screenshot of a computer

Description automatically generated

Ser access token before test it using postman

<https://www.youtube.com/watch?v=c-QsfbznSXI&t=3848s>