Settings.py)

from datetime import timedelta

from dotenv import load\_dotenv

import os

load\_dotenv()

ALLOWED\_HOSTS = ["\*"]

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),

}

INSTALLED\_APPS = [

    'api',

    'rest\_framework',

    'corsheaders',

CORS\_ALLOW\_ALL\_ORIGINS = True

CORS\_ALLOWS\_CREDENTIALS = True

Serializers.py)

class UserSerializer(serializers.ModelSerializer):

    class Meta:

        model = User

        fields = ['id','username','password']

**//This tells the serializer that we don’t want to return the password field but only want to store this field**

        extra\_kwargs = {'password':{'write\_only':True}}

class PostSerializer(serializers.ModelSerializer):

    class Meta:

        model = post

        fields = ['id','user\_name','query','content','time']

**//This is the opposite of before only allows us to read but cant write in it**

        extra\_kwargs = {'user\_name':{'read\_only':True}}

Project’s url.py)

from django.contrib import admin

from django.urls import path,include

from api.views import createUser

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

urlpatterns = [

    path('admin/', admin.site.urls),

    path('api/user', createUser.as\_view()),

    path('api/token', TokenObtainPairView.as\_view()),

    path('api/token/refresh', TokenRefreshView.as\_view()),

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

]

Views.py)

from django.shortcuts import render

from .serializers import UserSerializer

from rest\_framework import generics

from rest\_framework.permissions import IsAuthenticated, AllowAny

from django.contrib.auth.models import User

# Create your views here.

class createUser(generics.CreateAPIView):

    queryset = User.objects.all()

    serializer\_class = UserSerializer

    permission\_classes = [AllowAny]

Creating and deleting posts)

Views.py)

class createPost(generics.CreateAPIView):

    serializer\_class = PostSerializer

    permission\_classes = [IsAuthenticated]

    def get\_queryset(self):

        user = self.request.user

        posts = post.objects.filter(user\_name=user)

        return posts

    def perform\_create(self, serializer):

        if serializer.is\_valid():

            serializer.save(user\_name = self.request.user )

class deletePost(generics.CreateAPIView):

    serializer\_class = PostSerializer

    permission\_classes = [IsAuthenticated]

    def get\_queryset(self):

        user = self.request.user

        posts = post.objects.filter(user\_name=user)

        return posts

serializer.py)

class PostSerializer(serializers.ModelSerializer):

    class Meta:

        model = post

        fields = ['id','user\_name','query','content','time']

        extra\_kwargs = {'user\_name':{'read\_only':True}}

**FRONTEND**

-npm create vite@latest frontend -- --template react

Create 3 subfolders in src components, pages and styles. Files api.js ,components.js and .env