```python
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import os
from flask import Flask,render_template,request,jsonify,redirect
import pyrebase
import warnings
import jwt
warnings.filterwarnings("ignore")

firebaseConfig = {
  "apiKey": "AIzaSyA6DB5LsOP3SZtmwZtELAvSO67bepWv8Hg",
  "authDomain": "ibm-natural-disaster.firebaseapp.com",
  "databaseURL": "https://ibm-natural-disaster-default-rtdb.firebaseio.com",
  "projectId": "ibm-natural-disaster",
  "storageBucket": "ibm-natural-disaster.appspot.com",
  "messagingSenderId": "289608247253",
  "appId": "1:289608247253:web:3781d7e4fed3cc9e947f5c",
  "measurementId": "G-TMHEYWJF1B"
}
firebase=pyrebase.initialize_app(firebaseConfig)
auth=firebase.auth()

app=Flask(__name__,template_folder="templates")

model=load_model("Disasters.h5")

@app.route("/")
def home():
    return render_template("home.html")

@app.route("/loginnn")
def beginn():
    return render_template("index.html")

@app.route("/uploading",methods=["GET","POST"])
def up():
    if request.method=="POST":
        f=request.files["image"]
        basepath=os.path.dirname(__file__)
        filepath=os.path.join(basepath,"uploads",f.filename)
        f.save(filepath)
        img=image.load_img(filepath,target_size=(64,64))
        print(img)
        x=image.img_to_array(img)
        x=np.expand_dims(x,axis=0)
        op=['Cyclone', 'Earthquake', 'Flood', 'Wildfire']
        pred=np.argmax(model.predict(x))
        return render_template("uploading.html",result=op[pred])
        # return send_file(op[pred], mimetype='image/gif')
        # return op[pred]

    if request.method=="GET":
            return render_template("uploading.html")

@app.route("/signup",methods=["get","post"])
def gg():
    if request.method=="GET":
        return render_template("signup.html")
    if request.method=="POST":
        emailid=request.form["emailid"]
        password=request.form["password"]
        conf_password=request.form["conpassword"]
        if password ==conf_password:
            user=auth.create_user_with_email_and_password(emailid,password)
            return  render_template("index.html", msg="successfully registered")
        else:
            return render_template("signup.html",error="password and conform password not match")





@app.route("/login",methods=["post"])
def loginn():

    emailid=request.form["emailid"]
    password=request.form["password"]
    # user=auth.create_user_with_email_and_password(emailid,password)
    try:
        user=auth.sign_in_with_email_and_password(emailid,password)
    except:
        return render_template("index.html",msg="failed to login")
    return redirect("/uploading")




port = os.getenv('VCAP_APP_PORT', '8080')

if __name__ =="__main__":
    app.secret_key = os.urandom(12)
    app.run(debug=True, host='0.0.0.0', port=port )

```