```
from flask import Flask,render_template,request,redirect,url_for
import cv2
import tensorflow as tf
from tensorflow.python.keras.models import load_model
import numpy as np
import os
from werkzeug.utils import secure_filename
app = Flask(__name__ , template_folder="template")
model = load_model(r"/Model Collection/disaster.h5")
print("loaded model from disk")

@app.route('/', methods=['GET'])
def index():
    return render_template('Home.html')
@app.route('/home', methods=['GET'])
def home():
    return render_template('Home.html')
@app.route('/intro', methods=['GET'])
def intro():
        return render_template('intro.html')
@app.route('/webcam', methods=['GET', 'POST'])
def predict():
    print("[INFO] starting video stream...")
    vs = cv2.VideoCapture(0)

    (W, H) = (None, None)



    while True:

        (grabbed, frame) = vs.read()
```

```
if not grabbed:
    break



if W is None or H is None:
    (H, W) = frame.shape[:2]



output = frame.copy()

frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
frame = cv2.resize(frame, (64, 64))
# frame = frame.astype("float32")
x = np.expand_dims(frame, axis=0)
result = np.argmax(model.predict(x), axis=-1)
index = ['Cyclone', 'Earthquake', 'Flood', 'Wildfire']
result = str(index[result[0]])
# print(result)
# result=result.tolist()

cv2.putText(output, "activity: {}".format(result), (10, 120), cv2.FONT_HERSHEY_PLAIN,
            1, (0, 255, 255), 1)
# playaudio("Emergency it is a disaster")
cv2.imshow("Output", output)
key = cv2.waitKey(1) & 0xFF

# if the `q` key was pressed, break from the loop
if key == ord("q"):
    break
```

```
    # release the file pointers
    print("[INFO] cleaning up...")
    vs.release()
    cv2.destroyAllWindows()
    return render_template("webcam.html")


@app.route('/file', methods=['POST', 'GET'])
def video():
    if request.method == 'POST':
        uploaded_file = request.files['file1']
        if uploaded_file.filename != '':
            vid_name = str(uploaded_file.filename)
            print(vid_name + "Uploaded_Succesfully")
            uploaded_file.save(uploaded_file.filename)
            vs = cv2.VideoCapture(vid_name)
            if (vs.isOpened() == False):
                print("Error opening video stream or file")

            (W, H) = (None, None)
            while True:
                (grabbed, frame) = vs.read()
                if not grabbed:
                    break
                if W is None or H is None:
                    (H, W) = frame.shape[:2]
                output = frame.copy()
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame = cv2.resize(frame, (64, 64))
                x = np.expand_dims(frame, axis=0)
```

```
            result = np.argmax(model.predict(x), axis=-1)
            index = ['Cyclone', 'Earthquake', 'Flood', 'Wildfire']
            result = str(index[result[0]])
            cv2.putText(output, "activity: {}".format(
               result), (10, 120), cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 255), 1)
            cv2.imshow("Output", output)
            key = cv2.waitKey(1) & 0xFF
            if key == ord("q"):
               break
         print("[INFO] cleaning up...")
         vs.release()
         cv2.destroyAllWindows()
   return render_template("file.html")


@app.route('/image', methods=['POST', 'GET'])
def image():
   resulttext = ''
   if request.method == 'POST':
      uploaded_file = request.files['imgfile']
      if uploaded_file.filename != '':
         img_name = str(uploaded_file.filename)
         print(img_name + "Uploaded Succesfully")
         uploaded_file.save(uploaded_file.filename)
         from keras.models import load_model
         from keras.preprocessing import image
         model = load_model("disaster.h5")  # loading the model for testing
         img = image.load_img(img_name, grayscale=False,
                              target_size=(64, 64))  # loading of the image
         x = image.img_to_array(img)  # image to array
         x = np.expand_dims(x, axis=0)  # changing the shape
```

```
        pred = model.predict_classes(x)  # predicting the classes
        index = ['Cyclone', 'Earthquake', 'Flood', 'Wildfire']
        result = index[pred[0]]
        resulttext = result
    return render_template('image.html', result_text=resulttext)


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000, debug=True)
```