Build Python Code

- Build flask file ‘app.py’ which is a web framework written in python for server-side scripting.
- App starts running when “ name ” constructor is called in main.
- Render template is used to return html file.
- “GET” method is used to take input from the user.
- “POST” method is used to display the output to the user.
- Importing Libraries

```
1
2   # TEAM ID : PNT2022TMID07081
3   from flask import Flask,render_template,request
4   # Flask-It is our framework which we are going to use to run/serve our application.
5   #request-for accessing file which was uploaded by the user on our application.
6   import operator
7   import cv2 # opencv library
8   from tensorflow.keras.models import load_model#to load our trained model
9   import os
10  from werkzeug.utils import secure_filename
```

app.py

```
from flask import Flask,render_template,request
import numpy as np
import os
import operator
import cv2
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import load_img, img_to_array
from werkzeug.utils import secure_filename

app = Flask(__name__,template_folder="templates")
model=load_model('gesture.h5')
print("Loaded model from disk")

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

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

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

@app.route("/index6")
def index6():
        return render_template("index6.html")


@app.route('/index',methods=['GET','POST'])
def launch():
   return render_template("index6.html")

@app.route('/predict',methods=['GET','POST'])
def predict():

   if request.method == 'POST':
      print('inside launch function')
      f=request.files['image']

      basepath=os.path.dirname(__file__)
      file_path=os.path.join(basepath,'uploads',secure_filename(f.filename))
      f.save(file_path)
      print('img saved successfully')
      print(file_path)
```

```
    cap=cv2.VideoCapture(0)
    while True:
        _, frame=cap.read()
        frame=cv2.flip(frame,1)

        x1=int(0.5*frame.shape[1])
        y1=10
        x2=frame.shape[1]-10
        y2=int(0.5*frame.shape[1])

        cv2.rectangle(frame,(x1-1,y1-1),(x2+1,y2+1),(255,0,0)),1
        roi = frame[y1:y2,x1:x2]

        roi=cv2.resize(roi,(64,64))
        roi=cv2.cvtColor(roi,cv2.COLOR_BGR2GRAY)
        _, test_image=cv2.threshold(roi,120,255,cv2.THRESH_BINARY)
        cv2.imshow("test",test_image)

        result = model.predict(test_image.reshape(1,64,64,1))
        print(result)
        prediction =
{'ZERO':result[0][0],'ONE':result[0][1],'TWO':result[0][2],'THREE':result[0][3],'FOUR':result[0][4],'FIVE':res
ult[0][5]}
        prediction=sorted(prediction.items(),key=operator.itemgetter(1),reverse=True)

        cv2.putText(frame,prediction[0][0],(10,120), cv2.FONT_HERSHEY_PLAIN,1,(0,255,255),1)
        cv2.imshow("frame",frame)

        image1=cv2.imread(file_path)
        if prediction[0][0]=='ONE':
            resized=cv2.resize(image1,(200,200))
            cv2.imshow("Fixed Resizing",resized)
            key=cv2.waitKey(3000)

            if(key & 0xFF) == ord("1"):
                cv2.destroyWindow("Fixed Resizing")

        elif prediction[0][0]=='ZERO':
            cv2.rectangle(image1,(480,170),(650,420),(0,0,255),2)
            cv2.imshow("Rectangle",image1)
            cv2.waitKey(0)
            key=cv2.waitKey(3000)

            if(key & 0xFF)==ord("0"):
             cv2.destroyWindow("Rectangle")

        elif prediction[0][0]=='TWO':
            (h,w,d)=image1.shape
            center=(w//2,h//2)
            M=cv2.getRotationMatrix2D(center,-45,1.0)
```

```
            rotated=cv2.warpAffine(image1,M,(w,h))
            cv2.imshow("OpenCV Rotation",rotated)
            key=cv2.waitKey(3000)
            if(key & 0xFF)==ord("2"):
                cv2.destroyWindow("OpenCV Rotation")

        elif prediction[0][0]=='THREE':
            blurred=cv2.GaussianBlur(image1,(11,11),0)
            cv2.imshow("Blurred",blurred)
            key=cv2.waitKey(3000)
            if(key & 0xFF)==ord("3"):
                cv2.destroyWindow("Blurred")

        else:
            continue

        interrupt=cv2.waitKey(10)
        if interrupt & 0xFF == 27:
            break

    cap.release()
    cv2.destroyAllWindows()

    return render_template("home.html")

if __name__ == "__main__":
        app.run(debug=True
```

DONE BY

TEAM ID : PNT2022TMID07081