Project development Phase - 4

Flask app building

```
# import the necessary packages
from flask import Flask,render_template,request
# Flask-It is our framework which we are going to use to run/serve our application.
#request-for accessing file which was uploaded by the user on our application.
import cv2 # opencv library
from tensorflow.keras.models import load_model#to load our trained model
import numpy as np
from gtts import gTTS #to convert text to speech
from skimage.transform import resize
import os
from keras.preprocessing import image
from playsound import playsound
'''
def playaudio(text):
    speech=gTTS(text)
    print(type(speech))
    speech.save("output1.mp3")
    playsound("output1.mp3")
    return
'''
app = Flask(__name__,template_folder="templates") # initializing a flask app
# Loading the model
model=load_model('aslpng1.h5')
print("Loaded model from disk")
vals = ['A', 'B','C','D','E','F','G','H','I']

#app=Flask(__name__,template_folder="templates")
@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('/upload', methods=['GET', 'POST'])
def predict():
        # Get a reference to webcam #0 (the default one)
        print("[INFO] starting video stream...")
        vs = cv2.VideoCapture(0)
        #writer = None
        (W, H) = (None, None)

# loop over frames from the video file stream
        while True:
            # read the next frame from the file
            (grabbed, frame) = vs.read()
            # if the frame was not grabbed, then we have reached the end
            # of the stream
            if not grabbed:
                break
            # if the frame dimensions are empty, grab them
            if W is None or H is None:
                (H, W) = frame.shape[:2]
            # clone the output frame, then convert it from BGR to RGB
            # ordering and resize the frame to a fixed 64x64
            output = frame.copy()
            #print("apple")
            img = resize(frame,(64,64,1))
            img = np.expand_dims(img,axis=0)
            if(np.max(img)>1):
                img = img/255.0
            result = np.argmax(model.predict(img), axis=-1)
            index=['A', 'B','C','D','E','F','G','H','I']
            result=str(index[result[0]])
            #print(result)
            #result=result.tolist()
            cv2.putText(output, "It indicates: {}".format(result), (10, 120),
cv2.FONT_HERSHEY_PLAIN,
                        2, (0,255,255), 1)
            #converts text to speech and plays the audio
            speech = gTTS(text = result, lang = 'en', slow = False)
            #speech=gTTS(text)
            print(type(speech))
            speech.save("text.mp3")
            os.system("start text.mp3")
            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("upload.html")

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