Project Development - Delivery Of Sprint-3

STEP 1: Now we will be building a Flask application that is used for building our UI which in backend can be interfaced to the model to get predictions. Flask application requires an HTML page for Frontend and a Python file for the backend which takes care of the interface with the model.

Language used : Python

Required Packages :

```
Import numpy as np
Import cv2
Import os
From  keras.models import load_model
From flask import flask, render_template, response
Import tensorflow as tf
From gtts import gtts #to convert text to speech
Global graph
Global writer
From skimage.transform import resize
```

App.py

```
from flask import Flask, Response, render_template
from camera import Video

app = Flask(__name__)
@app.route('/')
def index():
        return render_template('index.html')

def gen(camera):
        while True:
                frame = camera.get_frame()
                yield(b'--frame\r\n'
                        b'Content-Type: image/jpeg\r\n\r\n' + frame +
                        b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
        video = Video()
        return Response(gen(video), mimetype='multipart/x-mixed-replace; boundary =
frame')
```

```
if __name__ == '__main__':
        app.run()
```

main.py

```
import cv2

video = cv2.VideoCapture(0)

while True:
        ret, frame = video.read()
        cv2.imshow("Frame", frame)
        k = cv2.waitKey(1)
        if k == ord('q'):
                break

video.release()
cv2.destroyAllWindows()
```

STEP 2: Each frame is taken from the camera and processed and sent to the model for prediction. As discussed image undergoes different processing steps to meet model requirements to get predictions.

Code to preprocess the frame captured from camera :

```
def detect(frame):
    img=resize(frame,(64,64,1))
    img=np.expand_dims(img,axis=0)
    if(np.max(img)>1):
        img=img/255.0
    with graph.as_default():
            prediction = model.predict_classes(img)
    print(prediction)
    pred=vals[prediction[0]]
```

camera.py

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

class Video(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)
        self.roi_start = (50, 150)
        self.roi_end = (250, 350)
        self.model = load_model('asl_model.h5') # Execute Local Trained Model
        # self.model = load_model('IBM_Communication_Model.h5') # Execute IBM
Trained Model
        self.index=['A','B','C','D','E','F','G','H','I']
        self.y = None
    def __del__(self):
        self.video.release()
    def get_frame(self):
        ret,frame = self.video.read()
        frame = cv2.resize(frame, (640, 480))
        copy = frame.copy()
```

```
                copy = copy[150:150+200,50:50+200]
                # Prediction Start
                cv2.imwrite('image.jpg',copy)
                copy_img = image.load_img('image.jpg', target_size=(64,64))
                x = image.img_to_array(copy_img)
                x = np.expand_dims(x, axis=0)
                pred = np.argmax(self.model.predict(x), axis=1)
                self.y = pred[0]
                cv2.putText(frame,'The Predicted Alphabet is:
'+str(self.index[self.y]),(100,50),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),3)
                ret,jpg = cv2.imencode('.jpg', frame)
                return jpg.tobytes()
```