```
# Creds Text to Speech
apikey = 'qoaM1vFcc9Vj7lbKKsZr97SnsPcYfb9ukR41BuEh900Z'
url = 'https://api.au-syd.text-to-speech.watson.cloud.ibm.com/insta


#setup service
authenticator = IAMAuthenticator(apikey)
#Create our service
tts = TextToSpeechV1(authenticator=authenticator)
#set the IBM service url
tts.set_service_url(url)

import ibm_db
try:
  conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=3883e7e4-18f5-4afe
  print("Successfully connected with db2")
except:
  print("Sorry.. Unable to connect : ", ibm_db.conn_errormsg())

app = Flask(__name__)


# Home page open aagum
@app.route('/')
def home():
  return render_template('home.html')



# register oda submit action
@app.route('/register',methods = ['POST', 'GET'])
def register():
  if request.method == 'POST':
    fname = request.form['fname']
    lname = request.form['lname']
    email = request.form['email']
    password = request.form['password']
```

```
    f.write(str(text))
    f.close
    with open('mode.txt', 'r') as f:
        text = f.readlines()
        text = ''.join(str(line) for line in text)
    with open('./winston.mp3', 'wb') as audio_file:
        res = tts.synthesize(text, accept='audio/mp3', voice='en-US_EmmaExpressive'
        audio_file.write(res.content) #writing the contents from text file to a aud

    playsound('winston.mp3')
    return render_template('txts.html')

@app.route('/asl',methods=['post'])
def asl():
    return render_template('asl.html')
@app.route('/object',methods=['post'])
def object():
    return render_template('object.html')
@app.route('/speech',methods=['post'])
def speech():
    return render_template('stxt.html')
@app.route('/aloud',methods=['post'])
def aloud():
    return render_template('txts.html')
@app.route('/about',methods=['post'])
def about():
    return render_template('about.html')

@app.route('/svr',methods=['post'])
def svr():
    return redirect("https://www.linkedin.com/in/siva-vimel-rajhen-05ab80194/")
@app.route('/ss',methods=['post'])
def ss():
    return redirect("https://www.linkedin.com/in/subiksha-sathiasai-9b3600195/")
@app.route('/ps',methods=['post'])
def ps():
    return redirect("https://www.linkedin.com/in/priyasha-s-23ba6121b/")
@app.route('/pks',methods=['post'])
def pks():
    return redirect("https://www.linkedin.com/in/praveen-kumar-2a8021218/")
@app.route('/github',methods=['post'])
def github():
    return redirect("https://github.com/IBM-EPBL/IBM-Project-45472-1660730304")


if __name__ == '__main__':
```

```
    sql = "SELECT * FROM user WHERE email =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    if account:
      return render_template('home.html', msg="You are alread
    else:
      if(len(password) < 6):
        return render_template('home.html', msg="Password sho
      else:
        insert_sql = "INSERT INTO user VALUES (?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, fname)
        ibm_db.bind_param(prep_stmt, 2, lname)
        ibm_db.bind_param(prep_stmt, 3, email)
        ibm_db.bind_param(prep_stmt, 4, password)
        ibm_db.execute(prep_stmt)
        return render_template('home.html', msg="Student Data
```

```
@app.route("/login", methods=["POST"])
def login():
  print("-------------------------")
  print("Inside login entrance")
  email = request.form.get("email")
  password = request.form.get("password")
  sql = "SELECT * FROM user WHERE email = ?"
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt, 1, email)
  ibm_db.execute(stmt)
  account = ibm_db.fetch_assoc(stmt)
  if not account:
    return render_template('home.html', msg="You are not yet
  else:
    print("+=================================+")
    print(account)
    print("+=================================+")
    print("Inside login")
    if(password == account['PASSWORD']):
      email = account['EMAIL']
      name = account['FNAME']
      lname = account['LNAME']
```

```
# Load Yolo
net = cv2.dnn.readNet("yolov3-coco/yolov3.weights", "yolov3-coco/yolo
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
outputlayers = [layer_names[i-1] for i in net.getUnconnectedOutLayers
colors = np.random.uniform(0, 255, size=(len(classes), 3))

# Loading image
cap = cv2.VideoCapture(0)


font = cv2.FONT_HERSHEY_PLAIN
starting_time = time.time()
frame_id = 0
while True:
    _, frame = cap.read()
    frame_id += 1

    height, width, channels = frame.shape

    # Detecting objects
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0

    net.setInput(blob)
    outs = net.forward(outputlayers)

    # Showing informations on the screen
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.2:
                # Object detected
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[3] * width)
                h = int(detection[2] * height)
```

```
model = keras.models.load_model("best_model_dataflair3.h5")
word_dict = {0:'A',1:'B',2:'C',3:'D',4:'E',5:'F',6:'G',7:'H',8:'I',9:'I
background = None
accumulated_weight = 0.5
ROI_top = 100
ROI_bottom = 300
ROI_right = 150
ROI_left = 350

def cal_accum_avg(frame, accumulated_weight):
    global background

    if background is None:
        background = frame.copy().astype("float")
        return None
    cv2.accumulateWeighted(frame, background, accumulated_weight)

def segment_hand(frame, threshold=25):
    global background

    diff = cv2.absdiff(background.astype("uint8"), frame)
    _ , thresholded = cv2.threshold(diff, threshold,255,cv2.THRESH_BINA
    image = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL, cv2
    contours,hierachy = cv2.findContours(thresholded.copy(), cv2.RETR_E

    if len(contours) == 0:
        return None
    else:
        # The largest external contour should be the hand
        hand_segment_max_cont = max(contours, key=cv2.contourArea)

        # Returning the hand segment(max contour) and the

        return (thresholded, hand_segment_max_cont)

cam = cv2.VideoCapture(0)
num_frames =0
while True:
    ret, frame = cam.read()

    frame = cv2.flip(frame, 1)
```

```python
import speech_recognition as sr
import pyttsx3

r = sr.Recognizer()

with sr.Microphone() as source:
    print("Listening...")
    r.pause_threshold = 1
    audio = r.listen(source,phrase_time_limit=5)
    print("Recognizing...")
    text = r.recognize_google(audio, language ='en-in')
    print(f"User said: {text.lower()}\n")
    f = open("mod.txt", 'w')
    f.write("Spoken : " + text)
    f.close
```