[4]:
```
import numpy as np
import face_recognition
import pyttsx3
import time
import cv2
import pytesseract
from gtts import gTTS
import os
import speech_recognition as sr
```

[6]:
```
def recognize_faces():
    # Initialize text-to-speech engine
    engine = pyttsx3.init()

    # Load known faces and names
    known_faces = ['photos/mohamed_elhalak.jpg']
    known_names = ['mohamed tarek']

    known_images = [cv2.imread(img) for img in known_faces]
    encodings_known = [face_recognition.face_encodings(img)[0] for img in known_images]

    # Initialize webcam
    cap = cv2.VideoCapture(0)
    face_announced = False

    while True:
        ret, img = cap.read()
        if not ret:
            print("Failed to capture frame from webcam")
            break

        img_resized = cv2.resize(img, (0, 0), None, 0.25, 0.25)
        img_resized = cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB)

        face_locations = face_recognition.face_locations(img_resized)
        encodings_current = face_recognition.face_encodings(img_resized, face_locations)

        if len(encodings_current) != 0:
            for face_encoding, face_location in zip(encodings_current, face_locations):
                matches = face_recognition.compare_faces(encodings_known, face_encoding)
                face_distances = face_recognition.face_distance(encodings_known, face_encoding)
                match_index = np.argmin(face_distances)

                if matches[match_index] and not face_announced:
                    name = known_names[match_index]
                    engine.say(name)
                    engine.runAndWait()
                    face_announced = True


                top, right, bottom, left = face_location
                top *= 4
                right *= 4
                bottom *= 4
                left *= 4
                cv2.rectangle(img, (left, top), (right, bottom), (0, 0, 255), 2)
                cv2.putText(img, name, (left + 6, bottom - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)

        else:
            face_announced = False

        cv2.imshow('Face Recognition', img)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

[ ]:

[7]:
```
def detect_objects():
    engine = pyttsx3.init()

    # Load class names
    with open('coco.names', 'rt') as f:
        classNames = f.read().rstrip('\n').split('\n')

    # Load model
    net = cv2.dnn_DetectionModel('frozen_inference_graph.pb', 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt')
    net.setInputSize(320, 230)
    net.setInputScale(1.0 / 127.5)
    net.setInputMean((127.5, 127.5, 127.5))
    net.setInputSwapRB(True)

    # Webcam capture
    cap = cv2.VideoCapture(0)
    detected_objects = set()
    last_announced_objects = set()

    while True:
        ret, img = cap.read()
        if not ret:
            break

        # Detect objects
        classIds, confs, bbox = net.detect(img, confThreshold=0.5)
```

```
        detected_objects_in_frame = set()

        # Process detections
        if len(classIds) != 0:
            for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):
                className = classNames[classId - 1]
                cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)
                cv2.putText(img, className, (box[0] + 10, box[1] + 20), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), thickness=2)
                detected_objects_in_frame.add(className)

                # Announce new objects in the frame
                if className not in detected_objects:
                    engine.say(f"{className}")
                    engine.runAndWait()

            # Announce objects that exited the frame
            for obj in detected_objects - detected_objects_in_frame:
                if obj not in last_announced_objects:  # Avoid duplicate announcements
                    engine.say(f"{obj} has left the frame.")
                    engine.runAndWait()
                    last_announced_objects.add(obj)

            # Reset last announced objects to track changes per frame
            last_announced_objects = detected_objects_in_frame.copy()

        detected_objects = detected_objects_in_frame

        # Display the resulting image
        cv2.imshow('Object Detection', img)

        # Break the loop if 'q' key is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release resources
    cap.release()
    cv2.destroyAllWindows()
```

[ ]:

[ ]:

[8]:

```
def recognize_text():
    while True:  # Loop until a valid input is provided
        # Prompt user for input
        mode = input("Enter 'live' for live camera input, 'static' for a static image, or 'pdf' for a PDF file: ").strip().lower()

        if mode == 'static':
            image_path = input("Please enter the path to the image file: ").strip()
            # Load image from the provided path
            frame = cv2.imread(image_path)
            if frame is None:
                print(f"Error loading image from {image_path}. Please check the file path.")
                continue  # Prompt for input again

        elif mode == 'live':
            # Delay for camera initialization
            time.sleep(2)
            cap = cv2.VideoCapture(0)
            time.sleep(2)  # Wait for the camera to be ready
            ret, frame = cap.read()
            cap.release()

            if not ret:
                print("Error capturing image from camera.")
                continue  # Prompt for input again

            # Optional: Add a delay before processing the image
            time.sleep(2)  # Delay to ensure the model can read the photo

        elif mode == 'pdf':
            pdf_path = input("Please enter the path to the PDF file: ").strip()
            # Open the PDF file
            doc = fitz.open(pdf_path)
            extracted_text = ""
            for page in doc:
                # Render each page to an image
                pix = page.get_pixmap()
                img = np.frombuffer(pix.samples, dtype=np.uint8).reshape(pix.height, pix.width, -1)
                img = cv2.cvtColor(img, cv2.COLOR_RGBA2BGR)  # Handle RGBA to BGR conversion
                # Recognize text in the rendered page
                extracted_text += pytesseract.image_to_string(img)
            doc.close()
            print("Extracted Text from PDF:", extracted_text)
            # Convert extracted text to audio
            tts = gTTS(text=extracted_text, lang='en')
            tts.save('output_audio.mp3')
            os.system('start output_audio.mp3')  # Adjust for OS if needed
            return  # End the function after processing the PDF

        else:
            print("Invalid option. Please enter 'live', 'static', or 'pdf'.")
            continue  # Prompt for input again

        # Recognize text in the captured or loaded frame
        extracted_text = pytesseract.image_to_string(frame)
        print("Extracted Text:", extracted_text)

        # Convert extracted text to audio
        tts = gTTS(text=extracted_text, lang='en')
        tts.save('output_audio.mp3')

        # Play the audio file
        os.system('start output_audio.mp3')  # Adjust for OS if needed
```

[10]:
```python
def recognize_speech():
    """Recognize speech from the microphone."""
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        audio = recognizer.listen(source)

        try:
            command = recognizer.recognize_google(audio)
            print("You said:", command)
            return command.lower()
        except sr.UnknownValueError:
            print("Sorry, I could not understand the audio.")
            return None
        except sr.RequestError as e:
            print(f"Could not request results from Google Speech Recognition service; {e}")
            return None
```

[19]:
```python
def main():
    print("Welcome! Would you like to (s)peak or (t)ype your commands?")
    input_method = input("Enter 's' for speech or 't' for typing: ").strip().lower()

    while True:
        if input_method == 's':
            command = recognize_speech()
        elif input_method == 't':
            print("""
            read? for -- 'reed: pdf,images,live_vid' --
            who?  for -- 'read:  faces' --
            what? for -- 'read: objects' --
            exit? for -- 'exit our project' --
            """)
            command = input("Please type wich model u need to run").strip().lower()
        else:
            print("Invalid option. Please enter 's' for speech or 't' for typing.")
            continue

        if not command:
            continue

        if any(word in command for word in ["voice", "read"]):
            recognize_text()
        elif any(word in command for word in ["face", "who"]):
            recognize_faces()
        elif any(word in command for word in ["object", "what"]):
            detect_objects()
        elif command == "exit":
            print("Exiting...")
            break
        else:
            print("Command not recognized. Please try again.")
```

[20]:
```python
main()
```

```
Welcome! Would you like to (s)peak or (t)ype your commands?
Enter 's' for speech or 't' for typing:  t

            read? for -- 'reed: pdf,images,live_vid' --
            who?  for -- 'read:  faces' --
            what? for -- 'read: objects' --
            exit? for -- 'exit our project' --

Please type wich model u need to run read
Enter 'live' for live camera input, 'static' for a static image, or 'pdf' for a PDF file:  pdf
Please enter the path to the PDF file:  pdf/sheet_pdf.pdf
Extracted Text from PDF: 21 ~ | &

A 8 c OO e F sc
‘Client Name Project Type ate Complete Hours Spent Amount Billec Hourly Rate
2 Karma Security Video Creation 6/30/2024 22 $ 1,100.00 $ 50.00
3 Elite Motors Proofreading 5/31/2024 2$ 120.00 $ 60.00
4 Sunshine Navig: Coaching 5/20/2024 14$ 742.00 $ 53.00
5 IceCap Producti Copy Editing 4/8/2024 1 $ 462.00 $ 42.00
© Pumpkin Naviga Ghostwriting 7/3/2024 8 $ 504.00 $ 63.00
7 Summit Electron Coaching 3/18/2024 33 $ 2,112.00 $ 64.00
8 — Grizzly Limited Ghostwriting 6/9/2024 14$ 630.00 $ 45.00
8 Thor Records Video Creation 71612024 23 $ 1,311.00 $ 57.00
10 Hurricane Netwc Ghostwriting 5/30/2024 20 $ 1,240.00 $ 62.00



            read? for -- 'reed: pdf,images,live_vid' --
            who?  for -- 'read:  faces' --
            what? for -- 'read: objects' --
            exit? for -- 'exit our project' --

Please type wich model u need to run who

            read? for -- 'reed: pdf,images,live_vid' --
            who?  for -- 'read:  faces' --
            what? for -- 'read: objects' --
            exit? for -- 'exit our project' --

Please type wich model u need to run wha
Command not recognized. Please try again.

            read? for -- 'reed: pdf,images,live_vid' --
            who?  for -- 'read:  faces' --
            what? for -- 'read: objects' --
            exit? for -- 'exit our project' --

Please type wich model u need to run what

            read? for -- 'reed: pdf,images,live_vid' --
            who?  for -- 'read:  faces' --
            what? for -- 'read: objects' --
            exit? for -- 'exit our project' --

Please type wich model u need to run exit
Exiting...
```

[ ]: