#Spoken to sign language

import sys

import cv2

import numpy as np

from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton, QLabel, QHBoxLayout, QVBoxLayout, QWidget, QSpacerItem, QSizePolicy

from PyQt5.QtGui import QFont, QPixmap, QImage

from PyQt5.QtCore import QTimer, Qt

import pyttsx3

import speech\_recognition as sr

import os

class SpeechRecognitionApp(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.setWindowTitle("Speech Recognition App")

        self.setGeometry(100, 100, 800, 800)  # Adjusted window size for image display

        self.main\_layout = QVBoxLayout()

        self.label = QLabel("Press the button and speak...", self)

        self.label.setFont(QFont("Arial", 12))  # Set font size to 12

        self.main\_layout.addWidget(self.label, alignment=Qt.AlignCenter)

        self.button = QPushButton("Start", self)

        self.button.clicked.connect(self.start\_recognition)

        self.main\_layout.addWidget(self.button, alignment=Qt.AlignCenter)

        self.image\_layout = QHBoxLayout()

        self.image\_widget = QWidget()

        self.image\_widget.setLayout(self.image\_layout)

        self.main\_layout.addWidget(self.image\_widget)

        self.video\_label = QLabel(self)

        self.video\_label.setFixedSize(500, 500)  # Set the size of the video display

        self.main\_layout.addWidget(self.video\_label, alignment=Qt.AlignCenter)

        central\_widget = QWidget()

        central\_widget.setLayout(self.main\_layout)

        self.setCentralWidget(central\_widget)

        self.engine = pyttsx3.init()

        self.recognizer = sr.Recognizer()

        self.timer = QTimer()

        self.timer.timeout.connect(self.update\_video\_frame)

        self.video\_frames = []

        self.current\_frame\_index = 0

    def start\_recognition(self):

        with sr.Microphone() as source:

            print("Listening...")

            self.label.setText("Listening...")

            self.label.repaint()

            audio = self.recognizer.listen(source)

            try:

                print("Recognizing...")

                self.label.setText("Recognizing...")

                self.label.repaint()

                text = self.recognizer.recognize\_google(audio)

                print("Recognized text:", text)

                self.label.setText(f"Recognized text: {text}")

                self.label.repaint()

                self.engine.say("Recognized text: " + text)

                self.engine.runAndWait()

                self.create\_video\_from\_text(text)

            except sr.UnknownValueError:

                print("Could not understand audio")

                self.label.setText("Could not understand audio")

                self.label.repaint()

            except sr.RequestError as e:

                print("Could not request results; {0}".format(e))

                self.label.setText(f"Could not request results; {e}")

                self.label.repaint()

    def create\_video\_from\_text(self, text):

        self.video\_frames = []

        self.current\_frame\_index = 0

        # Convert text to lower case and filter out non-alphabet characters

        text = ''.join(filter(str.isalpha, text.lower()))

        for char in text:

            image\_path = f"letters/{char}.jpg"

            if os.path.exists(image\_path):

                frame = cv2.imread(image\_path)

                if frame is not None:

                    frame = cv2.resize(frame, (300, 300))  # Resize frame to 500x500

                    self.video\_frames.append(frame)

        if self.video\_frames:

            self.timer.start(100)  # Update frame every 100ms

    def update\_video\_frame(self):

        if self.video\_frames:

            frame = self.video\_frames[self.current\_frame\_index]

            height, width, channel = frame.shape

            bytes\_per\_line = 3 \* width

            q\_img = QImage(frame.data, width, height, bytes\_per\_line, QImage.Format\_RGB888)

            self.video\_label.setPixmap(QPixmap.fromImage(q\_img))

            self.current\_frame\_index = (self.current\_frame\_index + 1) % len(self.video\_frames)

if \_\_name\_\_ == "\_\_main\_\_":

    app = QApplication(sys.argv)

    window = SpeechRecognitionApp()

    window.show()

    sys.exit(app.exec\_())