قم بالمراجعة الاكواد واعطيني فكرة تحسين وتطوير الملاحظات واجهة استخدام التطبيق

التركيز علي النموذج المحلي هل هناك نموذج افضل

التركيز علي نقطة طريقة التطوير والتحديث

بينما عند اخر الصفحة هناك نقاط مفترحة من رقم 4 وما بعد سنهتم به فيما بعد

والان كل ما عندي داخل المجلد هو كلاتي :

داخل المجلد C:\Users\bassam\Desktop\mine\_project

1. main.py
2. adaptive\_learner.py
3. task\_manager.py
4. voice\_interaction.py
5. speech\_recognition.py
6. requirements.txt
7. vosk-model-ar-0.22-linto-1.1.0
8. check\_voices.py
9. event\_handler.py
10. knowledge\_base.json
11. smart\_assistant.py

1.(main.py) هنا اصبح طلع شكل الايقونة ومربع الحوار ممتاز ونريد تطويرة ومراجعة باقي الاكواد

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QPushButton, QTextEdit, QLineEdit, QFrame, QVBoxLayout, QWidget

from PyQt5.QtGui import QPixmap, QFont

from PyQt5.QtCore import Qt, QTimer

from voice\_interaction import VoiceInteraction

from speech\_recognition import SpeechRecognizer  # استيراد مكتبة التعرف على الكلام

class AIAssistant(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.initUI()

        self.voice\_interaction = VoiceInteraction()

        self.welcome\_message()

        self.sound\_enabled = True  # تفعيل الصوت افتراضيًا

    def initUI(self):

        self.setGeometry(100, 100, 100, 100)

        self.setWindowFlags(Qt.FramelessWindowHint | Qt.WindowStaysOnTopHint | Qt.WA\_TranslucentBackground)

        self.setAttribute(Qt.WA\_TranslucentBackground)

        self.setWindowTitle('')

        central\_widget = QWidget(self)

        central\_widget.setAttribute(Qt.WA\_TranslucentBackground)

        self.setCentralWidget(central\_widget)

        self.chat\_icon = QLabel(central\_widget)

        pixmap = QPixmap("C:/Users/bassam/Desktop/mine\_project/Pngtree—social icon\_4421694.png")

        scaled\_pixmap = pixmap.scaled(100, 100, Qt.KeepAspectRatio)

        self.chat\_icon.setPixmap(scaled\_pixmap)

        self.chat\_icon.setAlignment(Qt.AlignCenter)

        self.chat\_icon.mousePressEvent = self.toggle\_chat\_frame

        layout = QVBoxLayout()

        layout.addWidget(self.chat\_icon)

        central\_widget.setLayout(layout)

        self.chat\_frame = QFrame(central\_widget)

        self.chat\_frame.setGeometry(0, 0, 300, 300)

        self.chat\_frame.setStyleSheet("background-color: rgba(255, 255, 255, 200); border: 1px solid gray; border-radius: 10px;")

        self.chat\_frame.hide()

        self.chat\_box = QTextEdit(self.chat\_frame)

        self.chat\_box.setGeometry(10, 10, 280, 150)

        self.chat\_box.setStyleSheet("background-color: rgba(255, 255, 255, 200); border: 1px solid gray; border-radius: 5px;")

        self.chat\_box.setReadOnly(True)

        input\_frame = QFrame(self.chat\_frame)

        input\_frame.setGeometry(10, 180, 280, 40)

        input\_frame.setStyleSheet("background-color: rgba(255, 255, 255, 200); border: 1px solid gray; border-radius: 5px;")

        self.input\_box = QLineEdit(input\_frame)

        self.input\_box.setGeometry(5, 5, 180, 30)

        self.input\_box.setStyleSheet("background-color: rgba(255, 255, 255, 200); border: 1px solid gray; border-radius: 5px;")

        self.send\_button = QPushButton("إرسال", input\_frame)

        self.send\_button.setGeometry(194, 5, 80, 30)

        self.send\_button.clicked.connect(self.send\_message)

        self.send\_button.setStyleSheet("background-color: lightblue; border: 1px solid gray; border-radius: 5px;")

        self.talk\_button = QPushButton("تحدث", self.chat\_frame)

        self.talk\_button.setGeometry(10, 228, 80, 30)

        self.talk\_button.clicked.connect(self.start\_voice\_interaction)

        self.talk\_button.setStyleSheet("background-color: lightgreen; border: 1px solid gray; border-radius: 5px;")

        self.update\_button = QPushButton("تحديث", self.chat\_frame)

        self.update\_button.setGeometry(100, 228, 80, 30)

        self.update\_button.clicked.connect(self.update)

        self.update\_button.setStyleSheet("background-color: yellow; border: 1px solid gray; border-radius: 5px;")

        self.close\_button = QPushButton("إغلاق", self.chat\_frame)

        self.close\_button.setGeometry(190, 228, 80, 30)

        self.close\_button.clicked.connect(self.close\_app)

        self.close\_button.setStyleSheet("background-color: red; border: 1px solid gray; border-radius: 5px;")

        self.sound\_button = QPushButton("إغلاق الصوت", self.chat\_frame)

        self.sound\_button.setGeometry(10, 265, 100, 30)

        self.sound\_button.clicked.connect(self.toggle\_sound)

        self.sound\_button.setStyleSheet("background-color: lightgray; border: 1px solid gray; border-radius: 5px;")

    def welcome\_message(self):

        """رسالة ترحيبية عند فتح التطبيق"""

        welcome\_text = (

            "مرحبًا! أنا ماني، مساعدك الذكي.\n"

            "حاليًا، الصوت العربي غير متوفر على نظامك.\n"

            "إذا كنت تريد سماع صوتي، يرجى إصلاح إعدادات الأصوات.\n"

            "حتى ذلك الحين، يمكنني التفاعل معك كتابيًا.\n"

            "كيف يمكنني مساعدتك اليوم؟"

        )

        self.add\_message(welcome\_text, "assistant")

        self.voice\_interaction.speak("Welcome! I am Manny, your smart assistant. Currently, Arabic voice is not available on your system. If you want to hear my voice, please fix the voice settings. Until then, I can interact with you textually. How can I help you today?")

    def toggle\_chat\_frame(self, event):

        if self.chat\_frame.isHidden():

            self.chat\_frame.show()

            self.setGeometry(100, 100, 320, 320)

        else:

            self.chat\_frame.hide()

            self.setGeometry(100, 100, 100, 100)

    def send\_message(self):

        message = self.input\_box.text()

        self.add\_message(message, "user")

        self.input\_box.clear()

        # الكشف عن لغة النص

        lang = self.voice\_interaction.detect\_language(message)

        print(f"لغة النص: {lang}")

        if lang == "en" and self.sound\_enabled:  # إذا كانت اللغة إنجليزية والصوت مفعل

            self.voice\_interaction.speak(f"You said: {message}")  # الرد بصوت إنجليزي

        response = self.voice\_interaction.respond(message)  # الرد كتابيًا

        self.add\_message(response, "assistant")

    def add\_message(self, message, sender):

        color = "green" if sender == "user" else "blue"

        self.chat\_box.append(f'<p style="color:{color}; border-radius: 5px; padding: 5px;">{message}</p>')

    def start\_voice\_interaction(self):

        """بدء التفاعل الصوتي"""

        # إنشاء كائن للتعرف على الكلام باستخدام المسار المطلق

        recognizer = SpeechRecognizer("C:/Users/bassam/Desktop/mine\_project/vosk-model-ar-0.22-linto-1.1.0")

        # الاستماع إلى الكلام

        print("يُرجى التحدث الآن...")

        try:

            text = recognizer.listen()

            print(f"تم التعرف على النص: {text}")

            # الكشف عن لغة النص

            lang = self.voice\_interaction.detect\_language(text)

            print(f"لغة النص: {lang}")

            if lang == "en" and self.sound\_enabled:  # إذا كانت اللغة إنجليزية والصوت مفعل

                self.add\_message(f"You said: {text}", "user")  # عرض النص الإنجليزي

                self.voice\_interaction.speak(f"You said: {text}")  # الرد بصوت إنجليزي

            else:  # إذا كانت اللغة عربية أو غير معروفة

                self.add\_message(f"لقد قلت: {text}", "user")  # عرض النص العربي

                response = self.voice\_interaction.respond(text)  # الرد كتابيًا بالعربية

                self.add\_message(response, "assistant")

        except Exception as e:

            print(f"حدث خطأ: {e}")

            self.add\_message("حدث خطأ أثناء التعرف على الكلام.", "assistant")

    def toggle\_sound(self):

        """تفعيل/إغلاق الصوت"""

        self.sound\_enabled = not self.sound\_enabled

        if self.sound\_enabled:

            self.sound\_button.setText("إغلاق الصوت")

        else:

            self.sound\_button.setText("تفعيل الصوت")

    def update(self):

        """تحديث التطبيق"""

        try:

            # تنزيل ملف التحديث الجديد

            update\_url = "http://example.com/path/to/update.zip"  # رابط ملف التحديث

            update\_path = "C:/Users/bassam/Desktop/mine\_project/update.zip"

            urllib.request.urlretrieve(update\_url, update\_path)

            # فك ضغط ملف التحديث وتطبيق التغييرات

            with zipfile.ZipFile(update\_path, 'r') as zip\_ref:

                zip\_ref.extractall("C:/Users/bassam/Desktop/mine\_project/")

            # حذف ملف التحديث بعد التحديث

            os.remove(update\_path)

            # إعادة تشغيل التطبيق

            self.restart\_app()

        except Exception as e:

            print(f"حدث خطأ أثناء التحديث: {e}")

            self.add\_message("حدث خطأ أثناء التحديث.", "assistant")

    def restart\_app(self):

        """إعادة تشغيل التطبيق"""

        QApplication.quit()

        QProcess.startDetached(sys.executable, sys.argv)

    def close\_app(self):

        self.close()

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

    assistant = AIAssistant()

    assistant.show()

    sys.exit(app.exec\_())

2.(adaptive\_learner.py)

import json

from pyarabic import araby

class AdaptiveLearner:

    def \_\_init\_\_(self):

        self.knowledge\_file = "knowledge\_base.json"

        self.knowledge = self.load\_knowledge()

    def load\_knowledge(self):

        try:

            with open(self.knowledge\_file, 'r', encoding='utf-8') as f:

                return json.load(f)

        except (FileNotFoundError, json.JSONDecodeError):

            return {

                "commands": {

                    "افتح المتصفح": {"action": "open\_browser", "program": None},

                    "احفظ الملف": {"action": "save\_file", "program": None}

                }

            }

    def save\_knowledge(self):

        with open(self.knowledge\_file, 'w', encoding='utf-8') as f:

            json.dump(self.knowledge, f, ensure\_ascii=False, indent=4)

    def learn\_command(self, command, action, program\_path=None):

        normalized = self.normalize\_text(command)

        self.knowledge["commands"][normalized] = {

            "action": action,

            "program": program\_path

        }

        self.save\_knowledge()

    def normalize\_text(self, text):

        """معالجة النص العربي بشكل متقدم"""

        text = araby.strip\_diacritics(text)  # إزالة التشكيل

        text = araby.normalize\_hamza(text)    # توحيد الهمزات

        return text.strip().lower()           # تحويل إلى حروف صغيرة

3.(task\_manager.py)

import subprocess

import psutil

class TaskManager:

    @staticmethod

    def open\_program(name):

        try:

            subprocess.Popen(name)

            return True

        except Exception as e:

            return str(e)

    @staticmethod

    def close\_program(name):

        try:

            for proc in psutil.process\_iter(['name']):

                if proc.info['name'] == name:

                    proc.kill()

            return True

        except Exception as e:

            return str(e)

4.(voice\_interaction.py)

الكود السابق

# في voice\_interaction.py

import pyttsx3

class ArabicTTS:

    def \_\_init\_\_(self):

        self.engine = pyttsx3.init()

        self.set\_arabic\_voice()

    def set\_arabic\_voice(self):

        voices = self.engine.getProperty('voices')

        for voice in voices:

            if 'arabic' in voice.name.lower():

                self.engine.setProperty('voice', voice.id)

                return

        raise Exception("تثبيت أصوات عربية من إعدادات النظام")

    def speak(self, text):

        self.engine.say(text)

        self.engine.runAndWait()

# الكود الاختباري يجب أن يكون خارج الفئة!

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

    tts = ArabicTTS()  # إنشاء الكائن هنا

    tts.speak("مرحبًا، أنا المساعد الذكي")  # استدعاء الدالة

الكود الحالي

  self.engine.say(text)

        self.engine.runAndWait()

    def detect\_language(self, text):

        """الكشف عن لغة النص"""

        try:

            lang = detect(text)

            print(f"لغة النص المكتشف: {lang}")  # طباعة لغة النص

            return lang

        except:

            return "unknown"

    def respond(self, text):

        """الرد على الرسالة"""

        lang = self.detect\_language(text)

        if lang == "en":

            return "You said: " + text

        else:

            return "لقد قلت: " + text

5.(speech\_recognition.py)

الكود السابق

from vosk import Model, KaldiRecognizer

import pyaudio

import json

class SpeechRecognizer:

    def \_\_init\_\_(self, model\_path):

        self.model = Model(model\_path)

        self.recognizer = KaldiRecognizer(self.model, 16000)

        self.p = pyaudio.PyAudio()

        self.stream = self.p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True, frames\_per\_buffer=8192)

    def listen(self):

        """الاستماع إلى الصوت وتحويله إلى نص"""

        print("ابدأ بالتحدث...")

        while True:

            data = self.stream.read(4096, exception\_on\_overflow=False)

            if self.recognizer.AcceptWaveform(data):

                result = json.loads(self.recognizer.Result())

                text = result.get('text', '')

                if text:

                    return text

الكود الحالي

class SpeechRecognizer:

    def \_\_init\_\_(self, model\_path):

        self.model = Model(model\_path)

        self.recognizer = KaldiRecognizer(self.model, 16000)

        self.p = pyaudio.PyAudio()

        self.stream = self.p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True, frames\_per\_buffer=8192)

    def listen(self):

        """الاستماع إلى الصوت وتحويله إلى نص"""

        print("ابدأ بالتحدث الآن...")

        while True:

            data = self.stream.read(4096, exception\_on\_overflow=False)

            if self.recognizer.AcceptWaveform(data):

                result = self.recognizer.Result()

                text = result.get('text', '')

                if text:

                    return text

6.(requirements.txt)

vosk

pyautogui

psutil

pyarabic

llama-cpp-python

pyaudio

pyttsx3

7.(vosk-model-ar-0.22-linto-1.1.0) مجلد وداخله

am

conf

graph

ivector

rescore

README

8.(check\_voices.py)

import pyttsx3

class MultiLangTTS:

    def \_\_init\_\_(self):

        self.engine = pyttsx3.init()

        self.voices = self.engine.getProperty('voices')

    def set\_voice\_by\_lang(self, lang='ar'):

        """تعيين الصوت بناءً على اللغة"""

        for voice in self.voices:

            if lang == 'ar' and ('ar' in voice.languages or 'arabic' in voice.name.lower()):

                self.engine.setProperty('voice', voice.id)

                print(f"تم تعيين الصوت العربي: {voice.name}")

                return

            elif lang == 'en' and ('en' in voice.languages or 'english' in voice.name.lower()):

                self.engine.setProperty('voice', voice.id)

                print(f"تم تعيين الصوت الإنجليزي: {voice.name}")

                return

        raise Exception(f"لم يتم العثور على صوت للغة: {lang}")

    def speak(self, text, lang='ar'):

        """توليف الكلام باللغة المحددة"""

        self.set\_voice\_by\_lang(lang)

        self.engine.say(text)

        self.engine.runAndWait()

9.(event\_handler.py)

class EventHandler:

    @staticmethod

    def on\_voice\_command(command):

        """معالجة الأوامر الصوتية"""

        pass

    @staticmethod

    def on\_generate\_code(task, instructions, previous\_code="", error=""):

        """توليد الكود"""

        pass

10.(knowledge\_base.json)

{

    "commands": {

        "افتح المتصفح": "open\_browser",

        "احفظ الملف": "save\_file",

        "اغلق المتصفح": "close\_browser"

    },

    "solutions": {

        "open\_browser": "TaskManager.open\_program('chrome.exe')",

        "close\_browser": "TaskManager.close\_program('chrome.exe')",

        "save\_file": "save\_file\_dialog"

    }

}

11.(smart\_assistant.py)

import tkinter as tk

from tkinter import ttk

class SmartAssistantUI:

    def \_\_init\_\_(self, master):

        self.master = master

        master.title("الذكي - واجهة التفعيل")

        master.geometry("300x400")

        # إنشاء نمط تصميم أنيق

        self.style = ttk.Style()

        self.style.configure('TButton', font=('Arial', 10), padding=5)

        # إطار رئيسي

        self.main\_frame = ttk.Frame(master, padding=20)

        self.main\_frame.pack(expand=True, fill='both')

        # عناصر الواجهة

        self.create\_widgets()

    def create\_widgets(self):

        # زر التفعيل

        self.activate\_btn = ttk.Button(

            self.main\_frame,

            text="تفعيل الذكي 🚀",

            command=self.activate

        )

        self.activate\_btn.pack(pady=10, fill='x')

        # زر التعلم التلقائي

        self.auto\_learn\_btn = ttk.Button(

            self.main\_frame,

            text="تعليم تلقائي 🤖",

            command=self.auto\_learn

        )

        self.auto\_learn\_btn.pack(pady=10, fill='x')

        # زر الأسئلة

        self.ask\_btn = ttk.Button(

            self.main\_frame,

            text="اسألني سؤال ❓",

            command=self.ask\_question

        )

        self.ask\_btn.pack(pady=10, fill='x')

        # زر التحقق من التعلم

        self.knowledge\_btn = ttk.Button(

            self.main\_frame,

            text="ماذا تعلمت؟ 🧠",

            command=self.show\_knowledge

        )

        self.knowledge\_btn.pack(pady=10, fill='x')

        # منطقة النتائج

        self.result\_label = ttk.Label(

            self.main\_frame,

            text="...أنا جاهز لمساعدتك",

            font=('Arial', 10),

            wraplength=250

        )

        self.result\_label.pack(pady=20)

    def activate(self):

        self.result\_label.config(text="✅ تم التفعيل بنجاح!\nأنتظر أوامرك...")

    def auto\_learn(self):

        self.result\_label.config(text="🔍 جاري البحث التلقائي...\nسيتم تحديث المعرفة تلقائيًا")

    def ask\_question(self):

        self.result\_label.config(text="📩 يمكنك طرح سؤالك الآن في حقل الدردشة")

    def show\_knowledge(self):

        self.result\_label.config(text="📚 المعرفة الحالية:\n- أساسيات الذكاء\n- تحليل البيانات\n- التعرف على الأنماط")

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

    root = tk.Tk()

    app = SmartAssistantUI(root)

    root.mainloop()

( نقاط بعض الخطوات المقترحة و هذا كل الاكواد في المجلد المطلوب)

1. تطبيق ذكي يتحدث معي ويتعلم بالغة العربي والانجيزي
2. هنا مشكلة نرجو معالجة المشكلة عند تاشير المواس علي الايقونه يجب طهر المربع واذا استمريت بالنقر علي المواس استطيع سحبه اي مكان في الشاشة وعندما يكون الواجهة صغير بمجرد انادي اسمه يفتتح المربع ويقول انا هنا واذا لم يسمع صوت يكتب ويقو تحدث انا استمع لك واذا لم اتكلم يصغر الشاشة يكون برنامج تفاعلي
3. هناك زر تحديث و نغير الاسم الي تطوير ونريد نركز علي هذا النقطة عندما بندا بتجهيز التطبيق ونحولة الي .exe سيكون برنامج مثبت ومفعل فقط يتحسن ويتطور من التحديثات
4. فيما بعد نستطيع تطوير واضافيات لغة برمجة ويستطيع يتطور حسب المطلوب ويتحكم بالاوامر وفتح البرامج والمتصفح ويستطيع التواصل معك
5. يستطيع يقرء ملفات بي دي اف ويعطيني تحليل الجدول بماذا مكون
6. ممكن اي معلومات ناقص يستطيع ان يتصفح وياخذ المعلومات الناقصة
7. يستطيع ان يتحكم بتحريك المواس وفتح البرامج علي سطح المكتب وفتح المتصفح
8. ممكن يسال بماذا اساعدك هل افتح لك تطبيقات مثل ..... كيف اعرف انه يعرف اسم البرنامج التي يقصدة او متصفح متخصص بي عندما يكون مفعل هو يتعلم وينسخ تجاه عملي ومنه يستنتج عناوين مسار فتح اي شي كنت اشتغل
9. ايضا اذا كان فعال وانا اسال deepseek او ذكاء اخر يستطيع ان يتعلم كيف التواصل معاهم لكي يقلتني في السوال وخذ المعلومات وينسخ الاكواد ويلصق الكود في برنامج python ويستطيع فتح محرر الاومر ويطبق مثل ما انا اطبق
10. ايضا ممكن نفعل زر تفعيل مستواء اعلي حيث اذا نقرنا علي التفعيل المستواء اعلي يفتتح بجانب مربع المحادثات 2 مربع محادثات بمجرد اتكلم او اكتب نص يتسجل في الثلاث المربع حيث الاول هو برنامجي التي يتحدث معي ويرد المربع الثاني هو التواصل مع deepseek ويتحدث معه كوسيط

كان برنامج deepseek مفتوح ولكن اراء كل الكلام في تطبيقي

والمربع الثالث نربط copilot الذكاء الصناعي الثالث موجود علي سطح مكتبي في شريط المهاء

1. الاهم من هذا يكون هناك زر للتحديثات اي اي تطوير له يتطور ويتحسن عندما نفعل له اضافات او نظلب من (ماني) صنع كود ليتطور منه