webchat

Provide the WebChat UI to use Azure AI Services.

Prerequisite

要在近端執行Webchat範例，需要先設定下列環境變數以連結Azure文字服務。

- `AZURE_LANGUAGE_ENDPOINT` =https://aitextdennis002.cognitiveservices.azure.com/
- `AZURE_LANGUAGE_KEY` =
- `AZURE_MY_QUESTIONANSWERING_PROJECT` =MyBookFastQnA
- `AZURE_QUESTIONANSWERING_ENDPOINT` =aitextdennis002.cognitiveservices.azure.com
- `AZURE_QUESTIONANSWERING_PROJECT` =aiqnadennis001
- `AZURE_SPEECH_ENDPOINT` =https://eastus.api.cognitive.microsoft.com/
- `AZURE_SPEECH_KEY` =
- `AZURE_SPEECH_REGION` =eastus
- `AZURE_TRANSLATE_DOC_ENDPOINT` =https://uicdennistranslate.cognitiveservices.azure.com/
- `AZURE_TRANSLATE_KEY` =
- `AZURE_TRANSLATE_REGION` =eastus
- `AZURE_TRANSLATE_TEXT_ENDPOINT` =https://api.cognitive.microsofttranslator.com
- `BOOKFAST_LANGUAGE_ENDPOINT` =https://bookfast-language-resource.cognitiveservices.azure.com/
- `BOOKFAST_LANGUAGE_KEY` =
- `BOOKFAST_QUESTIONANSWERING_PROJECT` =BookFast-Customer-KM

範例執行

本範例採用Flask設計，欲啟動範例，請使用下列命令:

```
(.venv) PS D:\Dennis\VSCode_Test\Flask\webchat> flask run
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [20/Oct/2025 11:22:15] "GET / HTTP/1.1" 200 -
```

透過瀏覽器存取`127.0.0.1:5000`，應該可以看到下列畫面:

功能說明

Azure QnA服務

透過`azure.ai.language.questionanswering`的`QuestionAnsweringClient`，使用`get_answers`方法，取得輸入問題的答案。其得到的答案準確性，會依據`Threshold`的設定而有所不同。

Azure文字轉語音服務

透過`azure.cognitiveservices.speech`服務，將得到的答案，使用內建三位的支援中文的語音資料，使用`SpeechSynthesizer`方法進行語音輸出。預設值沒有套用語音，需勾選`啟用語音`後，點擊`更新設定`按鈕套用。

Azure語音轉文字服務

一樣是透過`azure.cognitiveservices.speech`服務，但是因為目前尚未得知直接使用麥克風輸入語音至Azure Speech服務，因此在前端透過javascript使用`MediaRecorder`取得音訊，然後將之封裝成WebM格式傳至後端。然後在後端使用`ffmpeg`套件將WebM格式轉成Opus格式，然後再將之轉換成Wav格式，最後才使用Wav音訊轉換成文字。不能否認的，這段大部分是使用Vibe Coding及反覆測試完成。

Azure情感分析

有鑑於問答系統可能未盡完美，當使用者無法取得有效之答案後，可能會情緒性的發問，因此在此系統也導入Azure情感分析。對於每次的發問，一律給予分析，倘若負面情緒過高，則使用Email通知。

Email通知服務

透過`email.mime`套件，可以輕易完成寄送郵件的目的，這邊是使用對話紀錄為內文的方式，因此格式有點簡易。

對話紀錄功能 (Conversation Log)

為了更進一步的服務改善，將所有對話紀錄下來再進行研究分析，是一個不錯的方法。因為Azure也提供對話摘要服務，因此本系統以Azure所需的對話紀錄格式進行紀錄，但是目前Azure摘要服務似乎需要收費，所以並未連結此服務，而僅有對話紀錄檔。

回饋系統服務

目前設計為`不滿意`的回饋會觸發Email通知服務，`滿意`的問答並未處理，畢竟對於**真實**的回饋，尚未思考如何落實或判斷。

BookFast類別

`BookFast`類別提供上述AI服務與功能的方法，倘若後端AI服務需要更改，僅需移植後端服務即可。

```python
class BookFast():
    def __init__(self, cfg:QnAConfig):
        self._cfg = cfg
        self._qna = QuestionAnsweringClient(cfg.endpoint_language,
cfg.credential_language)
        self._feeling = TextAnalyticsClient(endpoint=cfg.endpoint_language,
credential=cfg.credential_language)
        self._summary = ConversationAnalysisClient(endpoint=cfg.endpoint_language,
credential=cfg.credential_language)

    # QnA
    def qna_session(self, message: str, threshold: int):
        return qna_session(self._cfg, message, threshold)

    # Feeling
    def sentiment_question(self, question: str):
        docs = [question]
        results = sentiment_analysis(self._feeling, docs)
        self._scores = sentiment_score(results[0])
        return sentiment_result(results[0])

    @property
    def scores(self):
        return self._scores

    # Email
    def send_email(self, message: str, to: str):
        # 建立郵件內容
        msg = MIMEText(f"""
        有一個新需求 =>
        {message}
        """)

        # 設定寄件者和收件者
        me = 'dennisliao66@gmail.com'
        you = to
        msg['Subject'] = 'BookFast FAQ notification'
        msg['From'] = me
        msg['To'] = you
        pwd = os.environ.get("GMAIL_APP_PASSWORD")
        # pwd = os.environ.get("OUTLOOK_APP_PASSWORD")

        # 建立 SMTP 連線
        with smtplib.SMTP(host="smtp.gmail.com", port=587) as smtp:
        # with smtplib.SMTP(host="smtp-mail.outlook.com", port=587) as smtp:
            try:
                smtp.ehlo()  # 驗證 SMTP 伺服器
                smtp.starttls()  # 建立加密傳輸
                smtp.login("dennisliao66@gmail.com", pwd)  # 登入寄件者 Gmail
                smtp.sendmail(me, [you], msg.as_string())  # 寄送郵件
```

```
                print("Complete!")
            except Exception as e:
                print("Error message:", e)

    # speech
    def text_to_speech(self, text: str, voice: str):
        speech_cfg = init_speech_config(self._cfg)
        # speech_cfg.speech_recognition_language="zh-TW"
        audio_cfg = speak_audiooutput_config()
        return text_to_speech(speech_cfg, audio_cfg, text, voice)

    def convert_to_wav(self, input_file, output_file):
        with open(input_file, 'rb') as f:
            data = f.read()

        # 假設原始資料是 PCM 格式
        # 你可能需要根據實際情況調整這些參數
        num_channels = 1
        sample_width = 2  # 16-bit
        # frame_rate = 128000 #44100
        frame_rate = 16000

        # 解析原始資料
        num_frames = len(data) // (num_channels * sample_width)

        # 創建 WAV 文件
        with wave.open(output_file, 'wb') as wf:
            wf.setnchannels(num_channels)
            wf.setsampwidth(sample_width)
            wf.setframerate(frame_rate)
            wf.writeframes(data)

    def wave_to_text(self, filename: str) -> str:
        speech_cfg = init_speech_config(self._cfg)
        # speech_cfg.speech_recognition_language="zh-TW"
        audio_cfg = file_audio_config(filename=filename)
        return speech_to_text(speech_cfg, audio_cfg)

    # 使用ffmpeg套件，仍然需要ffmpeg執行檔，
    # 因此若沒有安裝ffmpeg執行檔，轉檔將有錯誤。
    def llama_webm_to_text(self, webm, debug=False):
        import ffmpeg

        opus = 'temp.opus'
        wave = 'llama.wav'

        # 將 WebM 檔案轉換成 OPUS 格式
        stream = ffmpeg.input(webm)
        stream = stream.output(opus, vcodec='copy', acodec='libopus', ab=128000)
        stream.run()
```

```
        # 將 OPUS 檔案轉換成 WAV 格式
        stream = ffmpeg.input(opus)
        stream = ffmpeg.output(stream, wave, ar="16k", ac=1, format="wav")
        stream.run()
        if not debug:
            os.remove(opus)

        text = self.wave_to_text(wave)
        if not debug:
            os.remove(wave)

        return text
```