派愛工五

學號 110502538 姓名 林芝嫻 系級 資工三 B

- 題目發想

寂寞如影隨形，平時沒有朋友和我聊天，父母也對我不聞不問，想要假裝自己有朋友陪伴，想感受被人關心的感覺。然而，在這片孤寂的土地，我發現了一絲希望的光芒，那就是劉晨鐘老師的網路與資料庫程式設計。或許這就是生命中的一種啟示，我決定自己寫一個網頁，尋找友人聊天的感覺。孤單的日子讓我更加堅強，我開始擁抱資料庫，並把它當成一種成長的契機。

我寫了一個線上即時聊天網頁，可以多人加入聊天室同時聊天，但是我並沒有朋友，還可以和線上機器人聊天，訓練自己的英文能力，簡直一舉多得，如果喜歡這個網頁，還能付出實際行動贊助，讓我感受到家庭的溫暖。

- 功能說明

在首頁可以輸入使用者的暱稱，建立自己的聊天室，假如有朋友的話也可以要輸入房間 ID，加入他的聊天室，也可以去無人區和機器人聊天練習英文，如果想要加入派愛 premium 的話，可以贊助並掃碼支付，您將會獲得我的友誼一份，讓您感受到我的熱情。

在首頁可以輸入使用者的暱稱，如果沒輸入暱稱則會顯示"你這老登 都叫你輸入暱稱了"。

輸入暱稱後，即可建立房間，或是加入現有房間，聊天都是即時發送的，會在訊息旁邊顯示現在日期及時間。

若是想要練習英文的話，可以去無人區，裡面會有聊天醜魚和您聊天。因為醜魚聊天並不是非常智能，所以您可能會被氣死。他有女朋友，別惹他，他超兇。

想加入派愛 premium 的話，可以輸入您的姓名、電子郵件和贊助金額，並掃描付款碼，將會有專人為您服務。

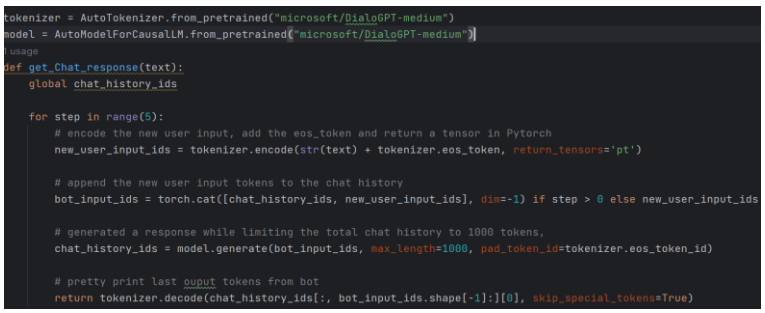
支付 QRcode 頁面。

- 實作技術

使用函式庫

```
from flask import Flask, render_template, request, session, redirect, url_for
from flask_socketio import join_room, leave_room, send, SocketIO
import random
from string import ascii_uppercase
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch
import sqlite3
```

這些程式碼定義了一個名為 get_Chat_response 的函數，使用了來自 Microsoft 的 DialoGPT-medium 模型，用於以對話方式生成回應。前兩行程式碼初始化了 tokenizer 和模型。Tokenizer 負責對文本進行編碼和解碼，而模型是一個針對對話任務設計的預訓練語言模型。然後定義了一個名為 **get_Chat_response** 的函數，接受一個文本輸入並生成回應。使用 tokenizer 對用戶的輸入進行編碼。準備模型的輸入並從模型生成回應使用 tokenizer 解碼生成的回應，跳過特殊標記。解碼的文本然後被返回。



```
tokenizer = AutoTokenizer.from_pretrained("microsoft/DialoGPT-medium")
model = AutoModelForCausalLM.from_pretrained("microsoft/DialoGPT-medium")
1 usage
def get_Chat_response(text):
    global chat_history_ids

    for step in range(5):
        # encode the new user input, add the eos_token and return a tensor in Pytorch
        new_user_input_ids = tokenizer.encode(str(text) + tokenizer.eos_token, return_tensors='pt')

        # append the new user input tokens to the chat history
        bot_input_ids = torch.cat([chat_history_ids, new_user_input_ids], dim=-1) if step > 0 else new_user_input_ids

        # generated a response while limiting the total chat history to 1000 tokens,
        chat_history_ids = model.generate(bot_input_ids, max_length=1000, pad_token_id=tokenizer.eos_token_id)

        # pretty print last ouput tokens from bot
        return tokenizer.decode(chat_history_ids[:, bot_input_ids.shape[-1]:][0], skip_special_tokens=True)
```

- 下面的程式碼是使用 Python 中的 Flask-SocketIO package，用於實現基於 WebSocket 的即時雙向通信。
- **@socketio.on("message") 函數:**

這個函數處理客戶端發送的 "message" 事件。當客戶端發送一條消息時，該函數會獲取當前用戶的房間（room）信息，檢查該房間是否存在。如果房間不存在，則不執行後續操作。否則，它創建一個包含發送消息的字典，然後使用 SocketIO 的 **send** 函數將消息發送給特定的房間。最後，它將消息添加到房間的消息列表中，並在伺服器端顯示消息。

- **@socketio.on("connect") 函數:**

這個函數處理客戶端的 "connect" 事件，表示有新的客戶端連接到伺服器。它檢查客戶端的房間和名稱是否存在，如果不存在，則返回。如果房間不存在，則從房間離開並返回。如果房間存在，則將客戶端加入該房間，發送一條歡迎消息給該房間，並更新房間中的成員數量。最後，它在伺服器端顯示用戶已經加入了哪個房間。

- **@socketio.on("disconnect") 函數:**

這個函數處理客戶端的 "disconnect" 事件，表示有客戶端斷開連接。它獲取斷開連接的客戶端的房間和名稱，然後從房間中離開。如果房間存在，則減少房間中的成員數量，如果成員數量小於或等於零，則從房間列表中刪除該房間。最後，它發送一條告別消息給該房間，並在伺服器端顯示用戶已經離開了哪個房間。

```python
@socketio.on("message")
def message(data):
    room = session.get("room")
    if room not in rooms:
        return

    content = {
        "name": session.get("name"),
        "message": data["data"]
    }
    send(content, to=room)
    rooms[room]["messages"].append(content)
    print(f"{session.get('name')} said: {data['data']}")
```

```python
@socketio.on("connect")
def connect(auth):
    room = session.get("room")
    name = session.get("name")
    if not room or not name:
        return
    if room not in rooms:
        leave_room(room)
        return

    join_room(room)
    send({"name": name, "message": "加入這個溫馨的大家庭"}, to=room)
    rooms[room]["members"] += 1
    print(f"{name} joined room {room}")
```

```python
@socketio.on("disconnect")
def disconnect():
    room = session.get("room")
    name = session.get("name")
    leave_room(room)

    if room in rooms:
        rooms[room]["members"] -= 1
        if rooms[room]["members"] <= 0:
            del rooms[room]

    send({"name": name, "message": "已離開 永遠懷念他"}, to=room)
    print(f"{name} has left the room {room}")
```

- 自我評估

完成這個專案時，遇到報錯問題訊息 **RuntimeError: The Werkzeug web server is not designed to run in production. Pass allow_unsafe_werkzeug=True to the run() method to disable this error** Flask 檢測到 Werkzeug 開發服務器在生產環境中使用時發出的警告。由於 Werkzeug 服務器是一個簡單的、單線程的服務器，不適合處理生產環境的復雜性和潛在的安全風險。程式中同時使用 flask-socketio 和 flask，觸發警告。解決方法是採用較舊版本的 Flask-SocketIO，我使用的版本是 5.2.0。

原本想使用 openai 的函式庫來用作聊天機器人的模型，發現需要購買使用額度遂改成使用 Microsoft 的 DialoGPT-medium 的模型，等到我獲得贊助支持，便會將聊天器人改版，讓各位使用者獲得更好的聊天體驗。

其他方面，除 UI 介面十分樸實無華外，網頁功能完善亦不會產生卡頓，個人認為這次專案學習到的網頁知識很多，可以了解到 socketio 的運作機制，還有將頁面和頁面之間的連動作好，還有其他參數傳遞需做的處理都能讓人獲益良多。

比較困難的地方是第一次接觸 flask-socketio，許多在前端需要提交的輸入文字和按鈕要傳回後端要多花時間了解，但學習過網路上的相關教學後，再完善整個專案就比較容易。本次專案是相當好的練習經驗，雖仍有美中不足的地方，但是一個很好的學習體驗。

本次專案 QRcode 連結。

140.115.197.133:8081