

Database Final Project

111550057莊婷馨、111550113謝詠晴、111550170卓建均、0811517楊雨宸

GitHub link: <https://github.com/vch2128/112-1-db-final-project>

Video link: <https://youtu.be/BfF6DXSHfz8>

Introduction

Motivation

近年來，學生的自殺率提升，心理健康問題有惡化的趨勢，交大這個學期也建立「心理假」的制度，可以了解學校同學非常需要心理方面的幫助。

由此發想，我們參考了並研究了市面上已存在的心理健康app (微日記、Moody、Moodpredd、DailyBean等等)，我們綜合了其有的共同功能，並加上額外的特色，發展出了具有記錄心情、心理疾病量表測驗、心理疾病介紹、以及心輔資源的linebot「心輔小幫手」。期許能夠透過簡單方便的聊天機器人模式，不但同學能對心理疾病有更多的認識，也能定期紀錄並檢視自己的心理狀態，並且在有需要的時候，能快速獲取心理輔導的相關資源。

Application Description

【LineBot: 心輔小幫手】

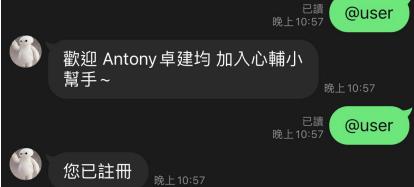


【6種功能選單】



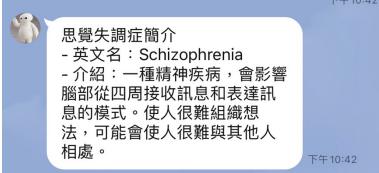
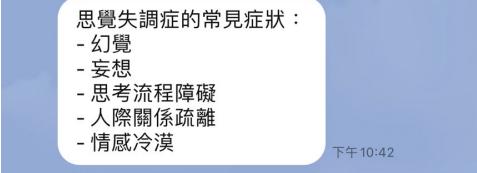
【會員註冊】

假如使用者第一次使用這個功能，會跳出歡迎(使用者名稱)加入心輔小幫手，若再點選會員註冊按鈕一次的話，則會顯示您已註冊的回覆。

step1. 點按會員註冊選單	step2. 回覆。再按一次將顯示您已註冊
	

【疾病介紹】

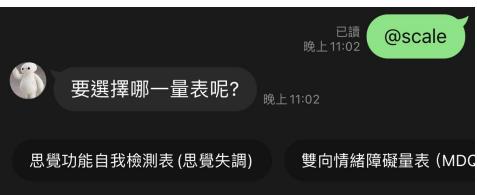
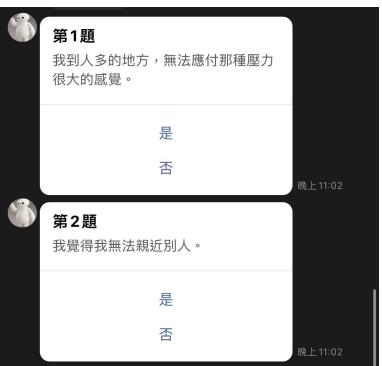
我們總共選擇了五種心理疾病：思覺失調症、雙向情緒障礙症、憂鬱症厭食症、注意力不足/過動症。點選疾病介紹的按鈕後，選擇要查看其中的哪個疾病資訊，疾病資訊包含：1.該疾病的簡介(英文名+介紹)、2.該疾病的常見症狀、3.該疾病在亞洲各國的盛行率。若選擇3，則跳出亞洲地區，分為東亞、東南亞、西亞、南亞和北亞，就能得知該地區中所有國家該疾病的盛行百分比。

step1. 點選疾病介紹選單	step2. 跳出不同疾病選單
	
step3. 點選簡介說明	step4. 點選常見症狀
	
step5. 點選各國盛行率(以東南亞為例)	 

【量表測驗】

我們分別上網搜尋了五種心理疾病的量表來給使用者做測驗，讓使用者能檢測出自己是否有可能患上心理疾病。點選量表測驗的按鈕後，會跳出五種表單可選擇想測試哪種疾病，點入想測試的表單後會從第一題的頁面開始跳出，每種表單問題形式都不同，有些用是或否來判斷，有些則用從不、偶爾、時常等頻率用詞來判斷，根據量表的結果我們設計讓選項在被選擇時會一邊統計分數，最後統計完總分後會跳出頁面，顯示測驗完成，總分為多少，然後給一些評語以及是否就醫的建議。

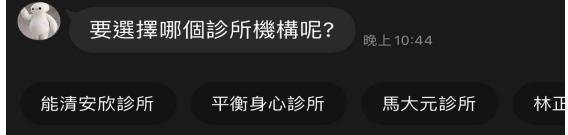
(使用範例)

step1. 選擇量表測驗按鈕	step2. 選擇表單後跳出題目，回答一題並計分後才會進入下一題
	
step3. 完成測驗，回報總分並給予建議	
	

【心輔資源】

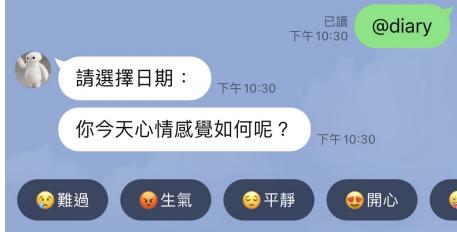
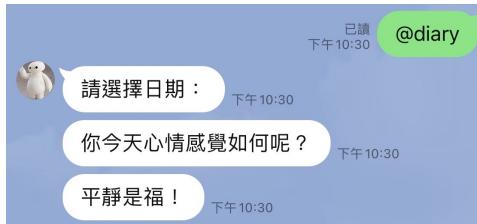
這個功能是能讓使用者依據自己所在的地區去查看該地區內的心理輔導機構(診所、醫院)有哪些，並透過google map能得知該診所在的地址，由於醫院診所眾多，我們這次只選擇了新竹縣市的四個地區來實作，分別是東區、北區、竹北市與竹東鎮。點選心輔資源按鈕後，會跳出四個地區的按鈕給使用者選擇，點下其中一個地區後會再跳出該地區所有的心理輔導機構供選擇，選擇想查看的機構之後會連接到google map標示出位置，並一同顯示出該機構的地址和電話，讓測試者能找到適合自己的心理輔導機構。

(使用範例)

step1. 點選心輔資源按鈕選擇想查看地區	step2. 選擇該地區的診所機構
	
step3. 跳出Google Map連結	step4. 點入連結查看詳細資訊
	

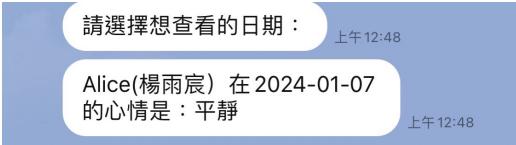
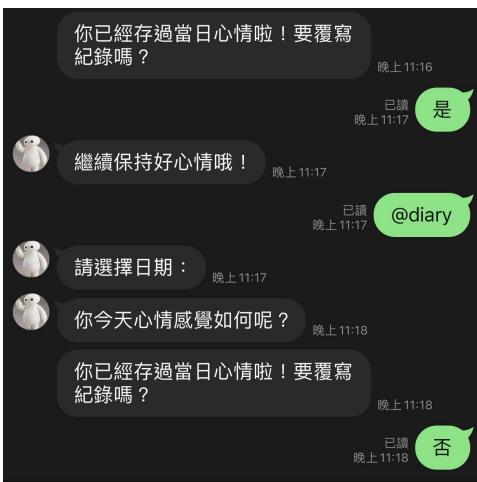
【心情日記】

我們設計了能讓使用者存下每日心情的功能，點入心情日記按鈕後，會跳出日期表(滾動式)給使用者選擇要存入哪一天的心情，選好日期後會跳出五種心情可選擇，分別是：難過、生氣、平靜、開心和興奮，選好該日的心情後會跳出該心情相對應的小語錄。另外我們還設計了另一個功能，假設使用者想要更改某天已經存入的心情，可以再次點選同一個日期，這次會跳出已經記錄過該日心情的訊息，然後會詢問使用者是否選擇覆寫，有是否兩個選擇，選是就會用SQL將資料做修正，選否則結束此功能。

step1. 點選選單中的心情日記	step2. 選取選日期
	
step3. 填入心情	step4. 紿予回饋
	

【數據紀錄】

這個功能是能告訴使用者單日、過去一周或是過去一個月的完整心情紀錄，能幫助使用者檢是一段時間內的心理狀況如何。點選數據紀錄的按鈕後，會跳出三種時間區段可選擇，若選擇查看單日的話，會跳出滾動式日曆給使用者選擇想查看的日期；若選擇查看過去一周的話，會依照五種心情被記錄的次數，列出該心情佔該周7天的比例(像是難過:2/7)，將五種心情的比例都秀出來；若選擇查看過去一個月的話則是將一整個月的心情紀錄調出，同樣是列出該心情佔該月30天的比例(像是難過:7/30)，讓使用者能隨時追蹤自己的心理狀態。

step1. 點選選單中的數據紀錄(選單日)	step2. 選取日期
	
step3. 顯示結果(結果1)	step3. 顯示結果(結果2)
	
回step2. 選取重複日期	step3. 顯示覆寫結果
	

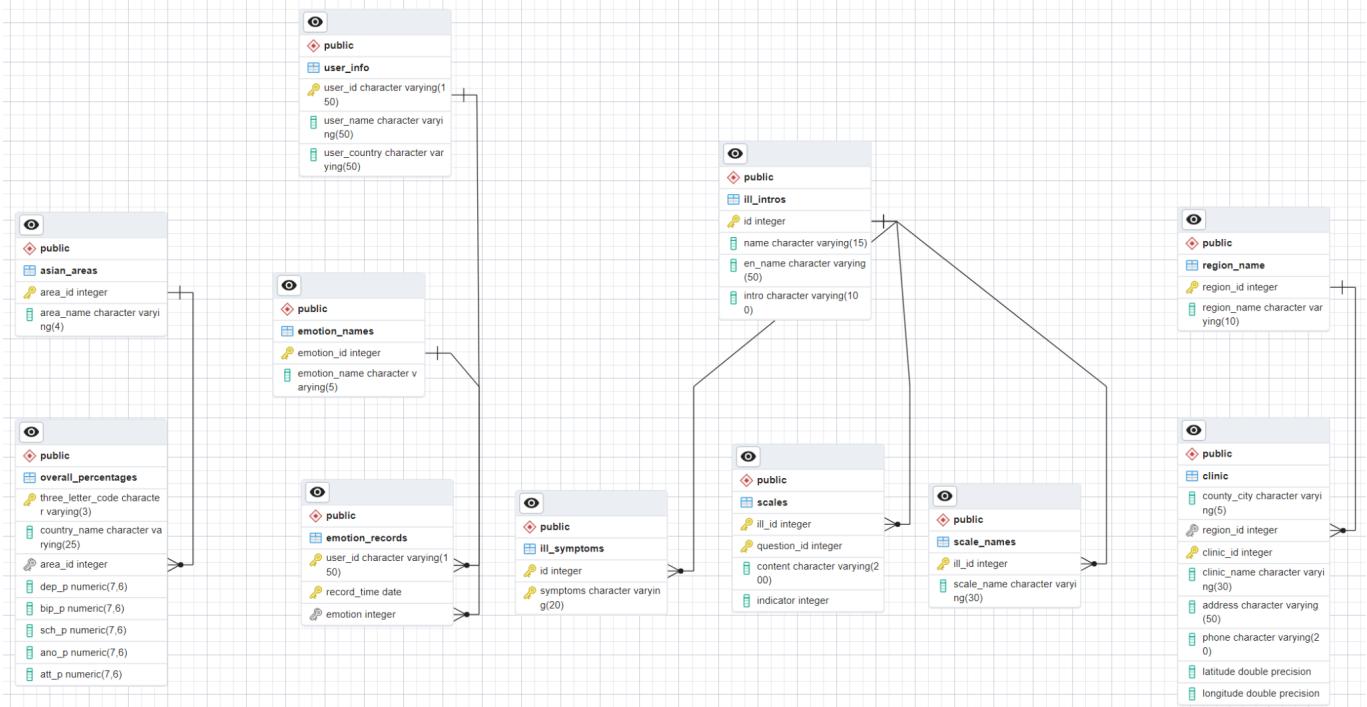
Database Design

SQL to create all the tables

```
1  create table emotion_names
2  (
3      emotion_id int,
4      emotion_name varchar(5),
5      primary key (emotion_id)
6  );
7  create table emotion_records
8  (
9      user_id varchar(150),
10     record_time date,
11     emotion int,
12     primary key (user_id, record_time),
13     foreign key (emotion) references emotion_names(emotion_id),
14     foreign key (user_id) references user_info(user_id)
15 );
16 create table test_emotion
17 (
18     emotion_id int,
19     emotion_name varchar(10),
20     primary key (emotion_id)
21 );
22 create table region_name
23 (
24     region_id int,
25     region_name varchar(10),
26     primary key (region_id)
27 );
28 create table user_id
29 (
30     user_id varchar(150),
31     user_name varchar(50),
32     user_country varchar(50),
33     primary key (user_id)
34 );
35 create table clinic
36 (
37     clinic_id int,
38     clinic_name varchar(30),
39     region_id int,
40     country_city varchar(5),
41     address varchar(50),
42     phone varchar(20),
43     latitude double precision,
44     longitude double precision,
45     primary key (clinic_id),
46     foreign key (region_id) references region_name(region_id)
47 );
48 create table ill_intro
49 (
50     id int,
51     name varchar(15),
52     en_name varchar(50),
53     intro varchar(100),
54     primary key (id)
55 );
```

```
56  create table ill_symptoms
57  (
58      id int,
59      symptoms varchar(20),
60      primary key (id, symptoms),
61      foreign key (id) references ill_intro(id)
62 );
63 create table scales
64 (
65     ill_id int,
66     question_id int,
67     content varchar(200),
68     indicator int,
69     primary key (ill_id, question_id),
70     foreign key (ill_id) references ill_intro(id)
71 );
72 create table scale_names
73 (
74     ill_id int,
75     scale_name varchar(30),
76     primary key (ill_id),
77     foreign key (ill_id) references ill_intro(id)
78 );
79 create table asian_areas
80 (
81     area_id int,
82     area_name varchar(4),
83     primary key (area_id)
84 );
85 create table overall_percentages
86 (
87     three_letter_code varchar(3),
88     country_name varchar(25),
89     area_id int,
90     dep_p numeric(7, 6),
91     bip_p numeric(7, 6),
92     sch_p numeric(7, 6),
93     ano_p numeric(7, 6),
94     att_p numeric(7, 6),
95     primary key (three_letter_code),
96     foreign key (area_id) references asian_areas(area_id)
97 );
```

Database Schema



Normal Form

我們的所有 table 都是 BCNF:

1. **emotion_names**
emotion_id 可以 uniquely define emotion_name (it's superkey for R)
2. **emotion_records**
(user_id,record_time) 可以 uniquely define emotion (it's superkey for R)
3. **test_emotion**
emotion_id 可以 uniquely define emotion_name (it's superkey for R)
4. **region_name**
region_id 可以 uniquely define region_name (it's superkey for R)
5. **user_id**
user_id 可以 uniquely define user_name, user_country (it's superkey for R)
6. **ill_intros**
id 可以 uniquely define name, en_name, intro (it's superkey for R)
7. **ill_symptoms**
(id, symptoms) 就是整個 table (it's superkey for R)
8. **scales**
(ill_id, question_id) 可以 uniquely define content, indicator (it's superkey for R)
9. **scale_names**
ill_id 可以 uniquely define scale_name (it's superkey for R)
10. **asian_areas**
area_id 可以 uniquely define area_name (it's superkey for R)

11. overall_percentages

three_letter_code 可以 uniquely define country_name, area_id, dep_p, bip_p, sch_p, ano_p, att_p (it's superkey for R)

12. clinic

clinic_id 可以 uniquely define clinic_name, region_name, region_id, country_city, address, phone, latitude, longitude (it's superkey for R)

由此可知我們的**table**都是在**BCNF**

Keys

- emotion_id 確保每一種情緒名稱有唯一標示
- (user_id, record_time) 確保每一位使用者在每天僅有一次情緒記錄
- region_id 確保每一個地區名稱有唯一標示
- user_id 確保每一個使用者擁有唯一的使用者 ID
- id 確保每一個疾病介紹是唯一的
- (id, symptoms) 確保每一個疾病症狀的組合是唯一的
- (ill_id, question_id) 確保每一個量表問題的組合是唯一的
- ill_id 確保每一個量表名稱對應到唯一的疾病介紹
- area_id 確保每一個亞洲的地區名稱有唯一標識
- three_letter_code 確保每一個國家的疾病盛行率(%)是唯一的
- clinic_id 確保每一家診所擁有唯一標識,

這樣的設計有助於提升查詢效率，能避免資料重複，並且讓資料庫更易於擴展。

Index

在pgAdmin 4中，創建primary key跟foreign key時，系統會自動在後台創建一個關聯的unique index，有助於加速相關的查詢操作。

Data Sources

Data Source and Original Format

依據來源，我們的資料可分為三類：

- kaggo
 - 亞洲國家心理疾病分佈資料(2019年)
 - 原本的格式包含包含不同性別、年齡區段、同國家不同年份的統計，還有亞洲以外的國家，由我們手動過濾並刪減，整理成 .csv 檔。
 - 來源網址
 - 思覺失調症、雙向情緒障礙症、憂鬱症 各國盛行率 來源：
<https://www.kaggle.com/datasets/amirhoseinmousavian/mental-health>

- 厥食症 各國盛行率 來源:
<https://www.kaggle.com/datasets/mpwolke/cusersmarildownloadsanamiacsv>
- 注意力不足/過動症 各國盛行率 來源:<https://vizhub.healthdata.org/gbd-results/>
- 手動搜集
 - 心輔資源診所
 - 使用搜尋引擎, 彙整診所基本資訊和經緯度成 .csv 檔
 - 各疾病測驗量表
 - 使用搜尋引擎, 彙整各疾病量表題目成 .csv 檔, 計分系統以 python 呈現
 - 疾病資訊
 - 使用搜尋引擎, 彙整疾病介紹和症狀成 .csv 檔
- 使用過程中搜集
 - 用戶資訊
 - 使用 LineBot 內建函式, 獲取用戶名稱和 ID, 並存入資料庫
 - 用戶心情日記記錄
 - 用戶可在 Line 介面中選取日期和當日心情, 輸入後存入資料庫, 作為「數據紀錄」的資料來源

Importing Data

根據資料來源, 有兩種方式:

- .csv 檔
 - 使用 “/copy” 指令, 從本機匯入資料
- 用戶輸入
 - 在 LineBot 中獲取用戶輸入的資訊後, 執行 SQL command, 透過 psycopg2 的連線輸入資料

Strategies for Updating Data

主要應用於「心情日記」的部分。

因為我們將 user ID 和日期作為 table 的 primary key, 所以一位用戶一天只能有一筆紀錄。當用戶輸入資料時:

- 若當日該用戶尚無資料, 更新該筆資訊至資料庫中。
- 若當日該用戶已經有資料, 詢問用戶是否覆寫資料。
 - 若是, 使用 query 更新原來的資料。
 - 若否, 不作變更。

Application with Database

Why Our Application Needs a Database

我們 LineBot 主要的功能分為兩大項：「查看心理疾病相關資訊」透過 query 獲取並篩選已建立的資料庫之資訊並呈現給使用者；「紀錄與回顧使用者每日心情」，藉由 LineBot 獲取使用者資訊及選擇的日期和心情，並 insert 至資料庫中，資料會不斷更新，以利使用者欲查看過去紀錄時能夠從中提取，以下根據六大功能分別說明：

- **疾病介紹：**

我們共有五種疾病的資訊，包含簡介說明文字、常見症狀及各國盛行率，使用者可點選不同按鈕查看，因此需要資料庫來儲存這些資料，以利能更快地利用 query 取得資訊。其中，在各國盛行率的部分，我們針對亞洲國家分成不同區域類別，更需要透過資料庫的 filter 來獲取該地區不同國家該疾病的盛行率。

- **量表測驗：**

我們提供五種疾病的自我評估量表，每一量表測驗有十數個題目，因此需要資料庫來處理這些大量題目資料，透過 query 和 primary key 的 id 讓使用者點選時，題目出現順序是正確的。

- **心輔資源：**

針對新竹縣市，蒐集不同行政區域的心理診所資訊，使用者先選擇某一行政區，我們會透過資料庫 query 列出該行政區的診所，使用者再點選欲查看之診所，一樣透過 query 取得該診所資訊，包含經緯度、地址等，使 LineBot 能以 Google Map 形式顯示。

- **心情日記：**

使用者先選擇一日期後，再選擇其當日的心情，為了存取其使用者、日期及心情的記錄，需要資料庫來儲存更新，會有新的資料不斷 insert 至資料庫中。

- **數據紀錄：**

在心情日記功能是能 insert 資料，而在數據紀錄則是可讓使用者回顧查看其過去紀錄的心情，我們分成單日、近七日、近一個月份，需要從使用者心情資料庫中透過 query 找到該使用者及日期所記錄的心情。

- **會員註冊：**

需要資料庫來存取全部有註冊 user 的 id 及名稱。

How to Perform Queries & Cooperating Functions

- **疾病介紹：**

- 取得五種疾病的id和名稱，使 LineBot 中的 Carousel button 顯示不同疾病名稱

```

SQL_ILL_INTRO_NAME = """
select id,name from ill_intro
"""

cur.execute(SQL_ILL_INTRO_NAME)           #fetch data from rds ill_intro table
conn.commit()
rows = cur.fetchall()
alldata = []
for row in rows:
    # Extract values from the row
    id, name = row
    values={"id":id,"name":name}
    alldata.append(values)

```

- [簡介說明]

透過 id 取得該疾病相對應的所有資訊，包含 id, name, en_name, intro

```

SQL_ILL_INTRO= """
select * from ill_intro
where id = %s
"""

cur.execute(SQL_ILL_INTRO,(int(item_id),))
conn.commit()
selected_row = cur.fetchall()
# Find the row with the matching item_id

```

而在 LineBot 中顯示的方式如下，因礙於篇幅且大部分功能按鈕的寫法相似，以下便不再贅述，詳細內容可查看程式碼檔案

```

if selected_row:
    id, name, en_name, intro = selected_row[0]
    msg = "%s簡介\n- 英文名 :%s\n- 介紹 :%s" % (name, en_name, intro)
    line_bot_api.reply_message(event.reply_token, TextSendMessage(text=msg))
else:
    line_bot_api.reply_message(event.reply_token, TextSendMessage(text="No matching data found"))

```

- [常見症狀]

取得該疾病相對應的所有症狀

```

SQL_ILL_SYMPTOM = """
select symptoms,name
from ill_symptoms
join ill_intro on ill_symptoms.id=ill_intro.id
where ill_symptoms.id = %s
"""

```

- [各國盛行率]

取得亞洲區域名稱，以在 button 列出可選擇的亞洲區域

```

SQL_AREA_LIST = """
select * from asian_areas
"""

```

藉 area_id 取得該地區所有國家不同疾病的罹患率數據

```
SQL_WANTED_AREA = """
select country_name,dep_p,bip_p,sch_p,ano_p,att_p
from overall_percentages
where area_id = %s
"""
```

藉 python if 判斷式及 sql select 的資料對應，將資訊顯示在對應的 button

```
cur.execute(SQL_WANTED_AREA,(int(area_id),))
conn.commit()
rows = cur.fetchall()
selected_row = []
for row in rows:
    country_name = row[0]
    perc = row[int(ill_id)]
    one_perc = '%s\n%s'%(country_name,perc)
    selected_row.append(one_perc)

cur.execute(SQL_ILL_INTRO_NAME)
rows = cur.fetchall()
for row in rows:
    if row[0]==int(ill_id):
        ill_name = row[1]

cur.execute(SQL_AREA_LIST)
result_set = cur.fetchall()
for row in result_set:
    if row[0]==int(area_id):
        area_name = row[1]

if selected_row:
    perc_str = "%\n- ".join(selected_row)
    msg = "%s在%s的盛行率:\n- %s%\n" % (ill_name,area_name,perc_str)
    line_bot_api.reply_message(event.reply_token, TextSendMessage(text=msg))
```

- 量表測驗：

- 取得各量表名稱，以顯示在 quick reply button 上

```
SQL_SCALE_NAME = """
select * from scale_names
"""

cur.execute(SQL_SCALE_NAME)
conn.commit()
rows = cur.fetchall()
```

- 依據使用者選擇的量表(scaletype)對應 ill_id，選擇 question_id，取得其對應問題內容文字。透過不斷 call 自己的 function 以跳出下一題。

```
SQL_SCALE_Q = """
select question_id,content,indicator from scales
where ill_id = %s and question_id=%s
"""

cur.execute(SQL_SCALE_Q,(int(scaletype),q_id,))
conn.commit()
rows = cur.fetchall()
# Find the row with the matching item_id
if rows:
    q_id, content, ind = rows[0]
```

- 心輔資源：

- 取得各地區名稱，以顯示在 button 上供使用者選擇

```
SQL_REGION = """
select * from region_name
""";
```

- 根據 region_id 取得該地區所有的clinic_id, clinic_name 以顯示在 quickreply button 上

```
sql="SELECT region_id,clinic_id,clinic_name FROM clinic"
cur.execute(sql)
result_set = cur.fetchall()
alldata=[]
for row in result_set:
    region_id,clinic_id,clinic_name = row
    if region_id==int(region_num):
        values={"clinic_id":clinic_id,"clinic_name":clinic_name}
        alldata.append(values)
```

- 根據使用者點選的 clinic, 取得該 clinic 的相關資訊, 並會顯示給使用者

```
sql="SELECT clinic_name,address,phone,latitude,longitude FROM clinic WHERE clinic_id=%s"%clinic_num
cur.execute(sql)
result_set = cur.fetchall()

if result_set:
    info = result_set[0]
    clinic_name,address,phone,latitude,longitude = info
```

- 心情日記：

- 檢查使用者是否已輸入過該日的心情

```
sql = "SELECT * FROM emotion_records WHERE user_id = %s AND record_time = %s;"
cur.execute(sql, (user_id, str(date)))
existing_record = cur.fetchone()
return existing_record is not None
```

- 若沒有輸入過, 則會 insert data 到 emtion_records 資料庫中

```
sql="INSERT INTO emotion_records (user_id,record_time,emotion) VALUES (%s, %s,%s);"
cur.execute(sql, (user_id, str(date),emotion_value))
conn.commit()
```

- 若已輸入過, 且使用者選擇要覆寫, 則會 update 該 user 該日的心情到資料庫

```
uid=event.source.user_id
profile=line_bot_api.get_profile(uid)
user_id=profile.user_id
sql = "UPDATE emotion_records SET emotion = %s WHERE user_id = %s AND record_time = %s;"
cur.execute(sql, (emotion_value, user_id, str(date)))
conn.commit()
```

- 數據紀錄：

- [查看單日]

- sql1 中根據 line_bot_api 獲取的 user_id 和使用者選擇的 date, 從資料庫中選取其該日的 emotion
- 在 sql2 中依據 emotion_id 得到對應的情緒名稱, 再顯示給使用者
- 若在資料庫中找不到資料, 則會顯示那天沒有紀錄

```

sql="SELECT record_time,emotion FROM emotion_records WHERE user_id=%s AND record_time=%s"
sql2="SELECT emotion_name FROM emotion_names WHERE emotion_id=%s"
cur.execute(sql,(user_id,str(date)))
rows=cur.fetchall()
msg = []
if rows:
    for row in rows:
        record_time,emotion=row
        cur.execute(sql2,(emotion,))
        emo=cur.fetchone()
        msg=user_name+"["+str(record_time)+"] 的心情是 ["+emo[0]
else:
    msg="您那天沒有記錄唷！"

```

- [查看一週內]

- 藉由 python 內建的 datetime 功能得到今日日期 current_time
- 在 sql 中藉由 user_id 及判斷 interval '7 days', 取得該使用者這七天內的所有心情紀錄
- 透過 emotion_id 和 join emtion_names, 取得心情對應名稱
- 顯示給使用者他這七天內, 各心情分別佔多少比例

```

SQL_SEVEN_DAYS = """
with aweek(user_id,record_time,emotion) as (
select * from emotion_records
where record_time > %s - interval '7 days' and user_id=%s
),
cnt(emo_id,cnt) as (
select emotion, count(*)
from aweek
group by emotion
)
select emotion_name,cnt
from cnt
join emotion_names on emotion_id=emo_id
"""

cur.execute(SQL_SEVEN_DAYS,(current_time,user_id,))
rows=cur.fetchall()
alldata = []
for row in rows:
    emo_name, cnt=row
    print(row)
    values = '%s[:] %s/7'%(emo_name,cnt)
    alldata.append(values)

```

- [查看一個月內] 同理一週內的方式, 將 interval '7 days' 改為 '30 days'

```

SQL_THIRTY_DAYS = """
with aweek(user_id,record_time,emotion) as (
select * from emotion_records
where record_time > %s - interval '30 days' and user_id=%s
),
cnt(emo_id,cnt) as (
select emotion, count(*)
from aweek
group by emotion
)
select emotion_name,cnt
from cnt
join emotion_names on emotion_id=emo_id
"""

```

- 會員註冊：

- 檢查使用者的資料是否已存在資料庫，若是，會顯示「您已註冊」

```
sql="SELECT COUNT(*) FROM User_info WHERE user_id = %s;"  
cur.execute(sql, (user_id,))  
count = cur.fetchone()[0]  
if(count>0):  
    line_bot_api.reply_message(event.reply_token, TextSendMessage(text='您已註冊'))
```

- 若沒有註冊過，則透過 line_bot_api 的 get_profile 取得使用者 id 和名稱，並 insert 至資料庫中

```
uid=event.source.user_id  
profile=line_bot_api.get_profile(uid)  
user_id=profile.user_id  
user_name=profile.display_name
```

```
sql = "INSERT INTO User_info (user_id,user_name) VALUES (%s, %s);"  
cur.execute(sql, (user_id,user_name))  
conn.commit()
```

Others

RDS

使用 AWS 的 RDS, 建立 PostgreSQL server

DB identifier	Status	Role	Engine	Region & AZ	Size	Recommendations
db-final-project	Available		Instance	PostgreSQL	us-east-1d	4 Informational

ngrok

使用 ngrok產生公開網址，透過筆電的 port 80，將 local 和 LineBot 連線。

- terminal 介面

```
127.0.0.1 -- [07/Jan/2024 18:01:30] "POST /callback HTTP/1.1" 200 -  
127.0.0.1 -- [07/Jan/2024 18:01:34] "POST /callback HTTP/1.1" 200 -  
127.0.0.1 -- [07/Jan/2024 18:14:31] "POST /callback HTTP/1.1" 200 -  
127.0.0.1 -- [07/Jan/2024 19:20:23] "POST /callback HTTP/1.1" 200 -  
127.0.0.1 -- [07/Jan/2024 19:29:16] "POST /callback HTTP/1.1" 200 -  
127.0.0.1 -- [07/Jan/2024 19:29:27] "POST /callback HTTP/1.1" 200 -  
127.0.0.1 -- [07/Jan/2024 19:29:31] "POST /callback HTTP/1.1" 200 -  
127.0.0.1 -- [07/Jan/2024 19:29:36] "POST /callback HTTP/1.1" 200 -
```

```

ngrok
Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status      online
Account            vch2128 (Plan: Free)
Version           3.5.0
Region            Japan (jp)
Latency          40ms
Web Interface    http://127.0.0.1:4040
Forwarding       https://cd67-140-113-136-220.ngrok-free.app -> http://127.0.0.1:4040

Connections        ttl     opn      r1l      rt5      p50      p90
                    244      0       0.00     0.00     0.77     0.98

HTTP Requests
-----
POST /callback      200 OK

```

- LineBot Webhook URL 介面

Webhook settings

Webhook URL ⓘ https://cd67-140-113-136-220.ngrok-free.app/callback

Verify **Edit**

(測試範例，非實際 URL)

Flask

Flask 是使用 python 撰寫的 Web 應用程式框架。使用以下架構開發我們的 LineBot:

```

from flask import Flask, request, abort

app = Flask(__name__)

@app.route("/callback", methods=['POST'])
def callback():
    global chat_button
    request_data = request.get_data(as_text=True)
    signature = request.headers['X-Line-Signature']
    body = request_data

    try:
        events = parser.parse(body, signature) # 傳入的事件
    except InvalidSignatureError:
        abort(400)
    except LineBotApiError:
        abort(400)

    # linebot events

    return "OK"

import os
if __name__ == "__main__":
    port = int(os.environ.get('PORT', 80))
    app.run(host='0.0.0.0', port=port)

```

line-bot-sdk

專門用來開發 LineBot 的 SDK。使用以下架構與 LineBot 連線：

```
from linebot import LineBotApi, WebhookParser,WebhookHandler
from linebot.exceptions import InvalidSignatureError, LineBotApiError
from linebot.models import *

#LineBot info stored in setting.py
try:
    line_bot_api = LineBotApi(LINE_CHANNEL_ACCESS_TOKEN)
    parser = WebhookParser(LINE_CHANNEL_SECRET)
    handler = WebhookHandler(LINE_CHANNEL_SECRET)
except:
    print("Error while connecting to LineBot")
```

psycopg2

用來執行與 PostgreSQL 資料庫相關的操作。使用以下架構與我們的資料庫連線：

```
import psycopg2

try:
    conn=psycopg2.connect(host=HOST_NAME,
                          user=USER_NAME,
                          password=PASSWORD,
                          dbname=DB_NAME,
                          port=PORT_NUM)
    cur=conn.cursor()
    print ("success")

except (Exception, psycopg2.Error) as error:
    print("Error while connecting to PostgreSQL:", error)

#LineBot functions

if conn:
    cur.close()
    conn.close()
    print("PostgreSQL connection is closed.")
```

未來展望

- 將本服務部署到雲端上，建立更加穩定且安全的運算措施
- 擴充心情日記功能，如：
 - 更多樣化的情緒紀錄
 - 一小段文字日記
- 擴充數據統計功能，使統計數字視覺圖表化
- 融合網路爬蟲，提供更多心輔資訊資訊