# Final Project

Project name: 空氣品質聊天機器人
Group members: 鍾昊哲 110550015、鄭旭閔 110550134、鄭博文 110550138

1. An introduction of your application, including why you want to develop the application and the main functions of your application.

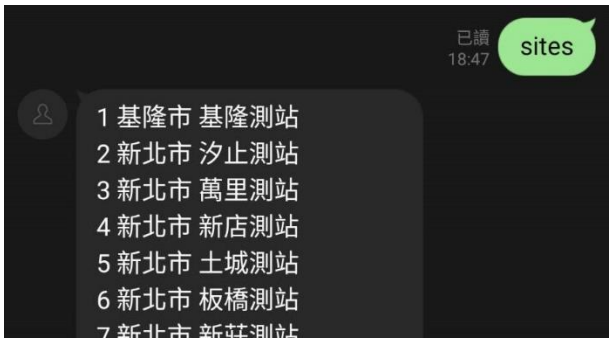
因工業、石化廢氣汙染，以及盛行季風帶來的霾害，我國大部分地區空氣品質長年不佳，威脅國人健康，國人對空氣品質的關注也因此與日俱增。

這次專案中，我們取用行政院環保署開放資料 API，作為資料來源。並以 LINE 聊天機器人(Line Chatbot) 為程式媒介，提供使用者對空氣品質的各式查詢、排程通知等功能，功能描述如下：

(1) 基本查詢

提供使用者特定測站的最新空氣品質資訊。系統保留一筆查詢紀錄，方便使用者快速查詢。包括 "sites", "<site id>", "last" 等指令，詳細使用方式可參考 demo 影片。

a. sites 指令：列出所有測站

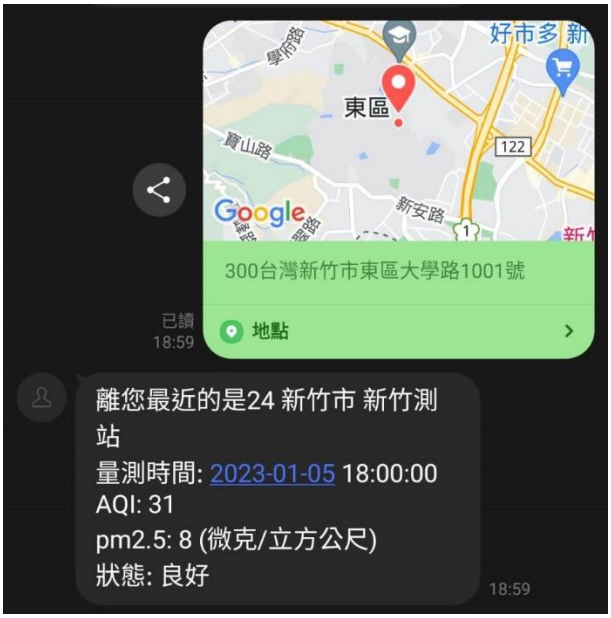



b. <siteId>指令：輸入數字可查詢該測站的最新資料

c. last 指令：取得上次查詢的測站的最新資料

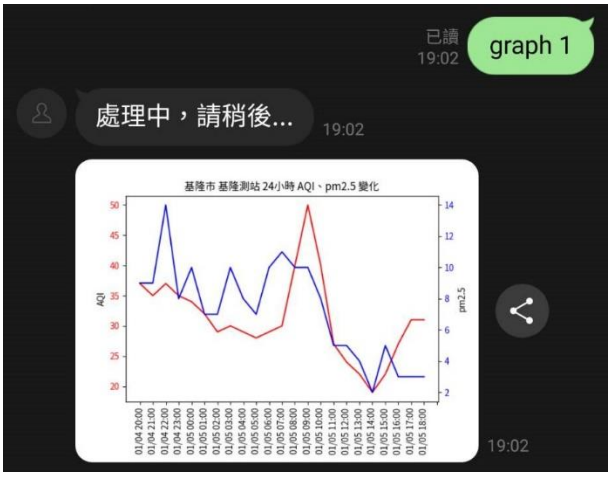

(2) 位置查詢

提供使用者以分享位置功能，查詢最近測站與最新資料。




(3) 圖表查詢

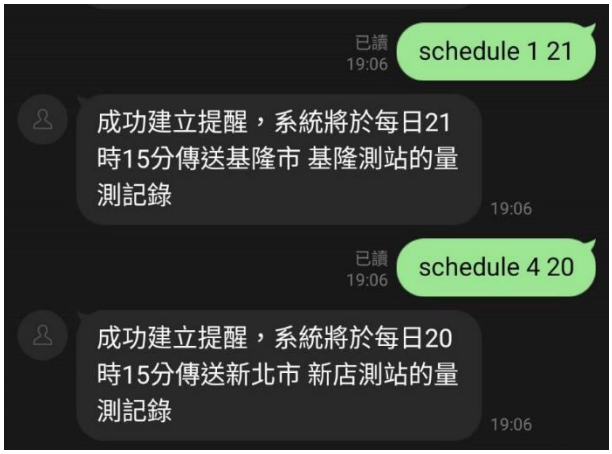
用"graph <siteId>"指令，提供使用者特定測站的 24 小時空氣品質圖表。



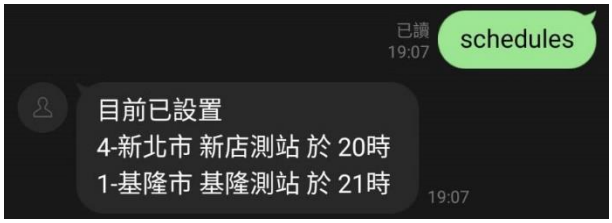
(4) 排程通知

提供使用者設定通知時間，由系統自動傳送測站資料。包括 "schedule", "schedules", "delete" 指令。

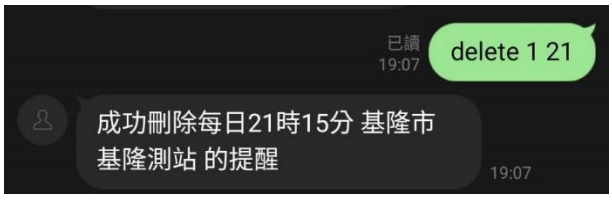
a. schedule <siteId> <hour>指令：在{hour}時 15 分傳送特定測站的資料。(因更新問題，使用者可以設定排程的小時，但排程的分鐘固定為 15)




b. schedules 指令：查看已設置的排程。

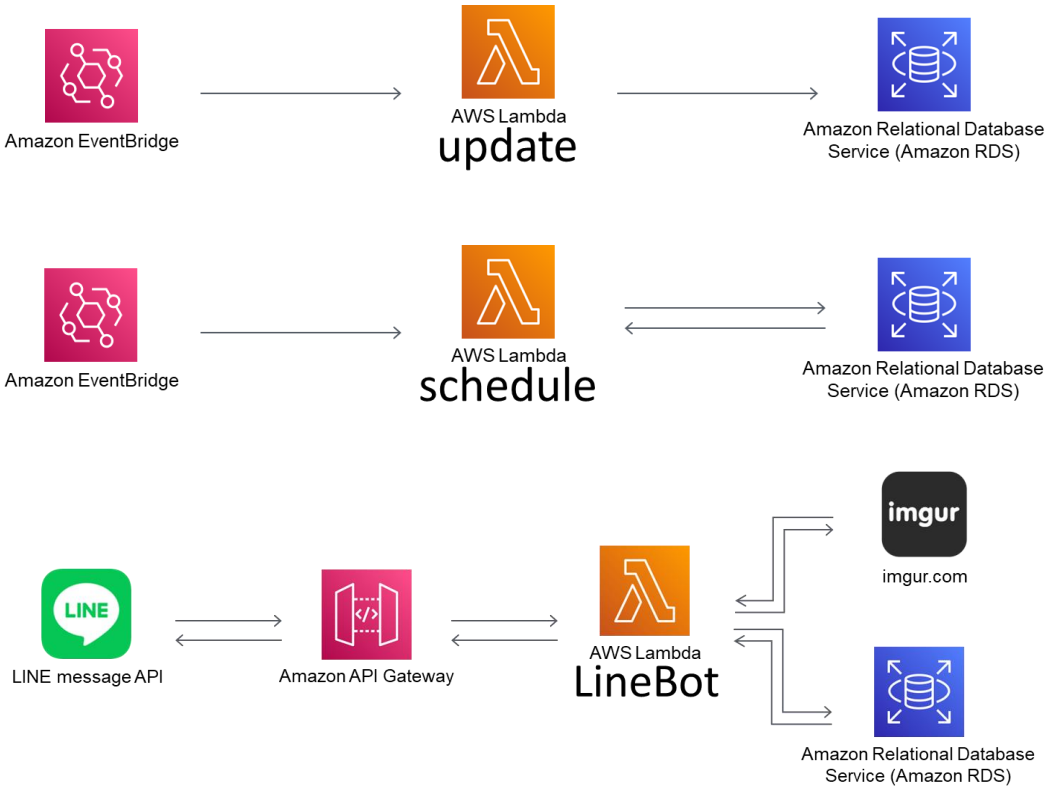



c. delete <siteId> <hour>指令：刪除排程




d. "schedule 4 20" 所產生的自動訊息(4-新店測站，寄送於 20 時 15 分)

此專案程式皆部署於 AWS 上，使用到 RDS, Lambda, API Gateway, Event Bridge 等服務。圖表查詢所用到的圖像資料則託管於 imgur.com 網站。架構如下：




2. Database design - describe the schema of all your tables in the database, including keys and index, if applicable (why you need the keys, or why you think that adding an index is or is not helpful).

(1) table - record

a. 此表用於儲存各測站各時間的空氣品質資料。資料來源為「行政院環保署開放資料 API」。

b. attributes, types and constraints

| attribute | type | constraint |
|---|---|---|
| siteId | smallint | primary key, foreign key references site |
| publishTime | timestamp | primary key |
| status | smallint | status >= 1 and status <= 6 |
| aqi | smallint | aqi >= 0 |
| pm2_5 | smallint | pm2_5 >= 0 |
| ⋮ | ⋮ | ⋮ |

c. 說明：

(a) primary key (siteId, publishTime)

(b) foreign key (siteId) references site

(c) status 表示空污狀況的分級，共六級，以數字表示。

(c)此表還留有更多空汙量值欄位，如二氧化硫、一氧化碳等，為後續功能擴展保留彈性，但因為現在還未投入使用，這邊就不贅述。

(2) table - site

a. 此表用於儲存各測站有關資料，如測站名、經緯位置等。

b. attributes, types and constraints

| attribute | type | constraint |
|---|---|---|
| siteId | smallint | primary key |
| siteName | varchar(10) | - |
| county | varchar(5) | - |
| longitude | numeric(7, 4) | longitude >= 0 and longitude <= 180 |
| latitude | numeric(6, 4) | latitude >= 0 and latitude <= 90 |

c. 說明：

(a) primary key (siteId)

(b)由於是我國測站，longitude 與 latitude 默認是東經與北緯。

(3) table - user_

a. 此表用於儲存 line 使用者有關資料，目前僅用來儲存上一筆查詢紀錄。系統也透過此表判斷使用者是否為初次使用。

b. attributes, types and constraints

| attribute | type | constraint |
|---|---|---|
| userId | char(33) | primary key |
| lastSiteId | smallint | foreign key references site |

c. 說明：

(a) primary key (userId)

(b) foreign key (lastSiteId) references site

(c) userId 為 line 使用者的內部 ID，由 base62 編成的 33 位獨特編號。須注意，此 ID 並非使用者自行設定，用來加好友的那種 ID。

(d) lastSiteId 用來記錄使用者上筆查詢，與 "last" 指令有關。

(e) table 名為"user_" (含尾綴底線)，用以區分管理資料庫使用者的內建 table。

(4) table - schedule

a. 此表用於儲存使用者排程通知有關的資料。

b. attributes, types and constraints

| attribute | type | constraint |
|---|---|---|
| userId | char(33) | primary key, foreign key references user_ |
| siteId | smallint | primary key, foreign key references site |
| time_ | smallint | primary key, time_ >= 0 and time_ <= 23 |

c. 說明：

(a) primary key (userId, siteId, time_)

(b) foreign key (userid) references user_

(c) foreign key (siteid) references site

(d) 排程提示時間為每小時十五分，time_ 用來紀錄提醒的時數，為 24 小時制，故值應為 0~23。(舉例：time_ = 20，則這筆提醒會於每日 20 點 15 分寄送)

(5) index - index_record_publishtime

a. 對 record table 中的 publishtime 做 index。

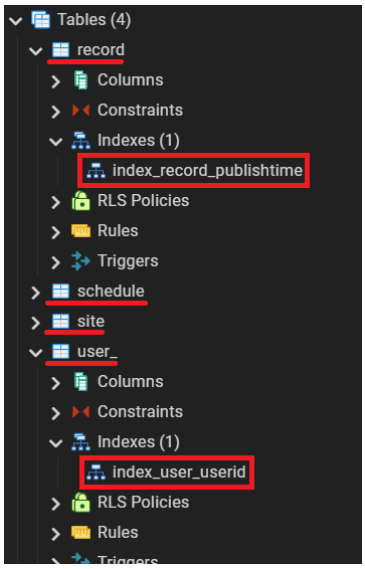
b. 原因：對 record 的 query 常常牽涉到 order by publishtime (詳見 7. (1) 之說明)

(6) index - index_user_userid

a. 對 user_中的 userid 做 index

b. 原因：由於每次使用者傳訊息都會牽涉到以 userid 查詢 row 的流程，所以在這裡加快操作。

(7)截圖：




(8)tables/indices SQL statements：請參考檔案 LineBot/db_design.sql。

3. Database design - describe the normal form of all your tables. If the tables are not in BCNF, please include the reason for it (performance trade-off, etc.).

本專案的 record、user_、schedule 內部皆沒有 non-trivial dependency，都是 BCNF。而 site 則具有 siteId -> siteName, county, longitude, latitude 以及 siteName -> siteId, county, longitude, latitude 的 dependency，但因為 siteId 與 siteName 都是 super key，所以也符合 BCNF。故四張 table 都是 BCNF。

4. From the data sources to the database - describe the data source and the original format.

(1)資料來源：行政院環保署開放資料 API
(2)資料說明網址：https://data.epa.gov.tw/dataset/detail/AQX_P_432
(3)API 請求網址：https://data.epa.gov.tw/api/v2/aqx_p_432 (須註冊會員，並提供 GET request parameter api_key)
(4)資料格式：JSON
(5)資料內容欄位名：

"sitename", "county", "aqi", "pollutant", "status", "so2", "co", "o3", "o3_8hr", "pm10", "pm2.5", "no2", "nox", "no", "wind_speed", "wind_direc", "publishtime", "co_8hr", "pm2.5_avg", "pm10_avg", "so2_avg", "longitude", "latitude", "siteid"

Data type 皆為字串。

5. From the data sources to the database - describe the methods of importing the original data to your database and strategies for updating the data, if you have one.

本專案只有 record table 需要定時更新，頻率是每小時。我們使用 Event Bridge 定時於整點十分(如 1:10、2:10、3:10 …)觸發更新用的 Lambda function。Lambda function 依照以下步驟進行更新，具體程式實作請參見程式碼 LineBot/grab.py。

(1) 用 python requests module，抓取前述的 API 網址。
(2) 爬取到的資料為 JSON 格式，故用 json module 分析資料。
(3) 用 psycopg2 module 連接 RDS 資料庫，將資料 insert 入 record table。

6. Application with database - explain why your application needs a database.

基於以下理由，我們必須使用資料庫。

(1) 資料來源端限制：

record table 的資料需要向環保署伺服器發請求，環保署 API 每日請求上限為 300 次，若每次使用者查詢都要進行請求，表示這個聊天機器人每日只能處理 300 次查詢。

另外，環保署 API 僅提供請求時的最新資料，即無法獲取過時的資料。但本專案的圖表查詢功能，需要過去一天，共計 24 份資料，這些資料必須由我方自行儲存在資料庫中。

(2)儲存需求：

本專案的查詢、設定排程等功能，都會需要調取/紀錄資訊，這些資訊需要長久保存，不能隨程式中止而消失。且這些資訊都有特定結構，適合存入資料庫。

7. Application with database - includes the queries that are performed by your application, how your application performed these queries (connections between application and database), and what is the cooperating functions for your application.

為了增強程式可讀性與系統安全性，本專案使用 psycopg2 與 sqlalchemy 這兩個 ORM module 進行資料庫連接與讀寫，有關資料庫的操作皆已函數化，存置於 LineBot/db.py，並於函數末以註解附上對應的 SQL statements。

(1) getLatestRecord(session, siteId)

作用：回傳給定測站的最新紀錄。

SQL：

**select * from record where siteid = {siteId}**

**order by publishtime desc limit 1;**

備註：排序後取末行並非查找最新(即 timestamp 最大) 紀錄的最佳實作法。但由於此作法通行，也被收錄於 SQLAlchemy 官方示範中，再加上有 index 加快效率，因此後續與「取最新」有關的 query 也都會循此方式實作。

(2) addUser(session, userId)

作用：新增使用者。

SQL：**insert into user_ values ({userId}, null);**

(3) getUser(session, userId)

作用：回傳使用者物件。

SQL：**select * from user_ where userid = {userId} limit 1;**

(4) getSite(session, siteId)

作用：回傳測站物件。

SQL：**select * from site where siteid = {siteId} limit 1;**

(5) getSites(session)

作用：回傳全部測站物件，以 siteId 排序。

SQL：**select * from site order by siteid;**

(6) existSite(session, siteId)

作用：回傳測站是否存在。

SQL：**select {siteId} in (select siteid from site);**

(7) getRecordsAfter(session, siteId, timeLimit)

作用：回傳特定測站、特定時間點以後的紀錄，以 publishtime 排序。

SQL：

```
select * from record
where siteid = {siteId} and publishtime >= {timeLimit}
order by publishtime;
```

(8) getSchedule(session, userId, siteId, hour)

作用：回傳排程物件。

SQL：

```
select * from schedule
where userid = {userId} and siteid = {siteId} and time_ = hour limit 1;
```

(9) addSchedule(session, userId, siteId, hour)

作用：新增排程。

SQL：**insert into schedule values ({userId}, {siteId}, {hour});**

(10) getScheduleWithSite(session, userId)

作用：回傳使用者的排程與對應測站物件。

SQL：

**select * from schedule as sc natural join site as si**

**where sc.userid = {userId} order by sc.time_;**

8. All the other details of your application that you want us to know.

(1) 這次專案更複雜，使用套件也更多，所以使用 lambda function 的 layer 來做套件管理。

(2) 套件中的 matplotlib 需要基於安裝於正常位置的 numpy，若需要使用這些套件，必須將 lambda 的 python 版本降至 3.7，使用 AWS 提供的 "AWSLambda-Python37-SciPy1x" layer。

(3) layer 使用的套件檔案龐大，且可自行基於環境安裝配置，就不特別上傳 e3。

(4) 以下補充說明現形架構設計原因：

a. 使用 EventBridge：

EventBridge 能定時產生事件，觸發其他 AWS 服務，所以需要定時執行的 update、schedule 兩個 Lambda function，透過 EventBridge 啟動。

b. 使用 API Gateway：

Line 的 message API 是以 POST request 的方式傳遞給我方 server，所以傳統上會以 flask 等輕量套件，寫出 server 程式。但由於 Line 規定我方 server 必須使用 https 憑證，所以我們改用 API Gateway，並用其調用地址作為 server 站址。API Gateway 接到請求後再轉給 LineBot Lambda function，作後續動作。

c. 使用 imgur.com：

LineBot 傳送圖片，必須先將圖片傳上網路，再將圖片 URL 傳給 Line server。由於 imgur.com 具有程式上傳 API，故將圖片託管於 imgur.com。