## A. Project code

**https://github.com/ptkuo0322/FinalProj_SI507.git**

## B. Data sources

### 1. YELP fusion API

- Original URL: https://api.yelp.com/v3/businesses/search
- Documentation: shorturl.at/loHP7
- Data format: JSON
- How to access the data
  - Ask user to key in the **RESTARUANT/FOOD** and **LOCATION** as the information
  - Pass the user's input with the API_key as the query term
  - Use HTTP. Request to retrieve the data from Yelp Fusion website
  - Parsed the searched result and save as json file (for cache) and csv file (for SQL database)
- Summary of data
  - # records available > 200 (estimated)
  - # records retrieved > 100 (estimated)
  - Name/phone/address will be retrieved to give the direct information about the restaurant. ReviewCount/rating will show the popularity. Latitude/longitude will show the relative location and the direction on maps. Id is the element used to identify the information in the SQL database
- Evidence of caching
  - shorturl.at/irtE0

### 2. YouTube API

- Original URL: https://www.googleapis.com/youtube/v3/videos
- Documentation: shorturl.at/kAIUY
- Data format: JSON
- How to access the data
  - Ask the users which searched restaurant they are interested in?
  - Pull out the restaurant name from the cache created from YELP query step and pass it as the query term to search on YouTube
  - Parse the searched result and save as json file and to show the related videos on web browser
- Summary of data
  - # records available > 200 (estimated)
  - # records retrieved > 100 (estimated)
  - Several central component of URL links of restaurants will be retrieved here. It will be first re-constructed in a specific format and use to search the videos on YouTube to give a glimpse of the restaurant.
- Evidence of caching
  - shorturl.at/mBEIL

### 3. Google static map API

- Original URL: https://maps.googleapis.com/maps/api/staticmap
- Documentation: shorturl.at/nrxCQ, shorturl.at/KNT17
- Data format: JSON
- How to access the data

    - Read the cache file created by the YELP query step to get the latitude/longitude
    - Re-construct the information and use as query key for static map API
    - Pass to static map API to get the location-marked static map on web browser
  - Summary of data
    - # records available > 200 (estimated)
    - # records retrieved > 100 (estimated)
    - The latitude and longitude attributes in the cache created by YELP query step will used to mark the relative locations of restaurants. Users are able to know the distance between each two of them (for better arranging their time and schedule)
  - Evidence of caching
    - shorturl.at/irtE0 (This one uses the cache created in Yelp query step)

4. Google map API
   - Original URL: https://www.google.com/maps/search/?api=1
   - Documentation: shorturl.at/jmnoP
   - Data format: JSON
   - How to access the data
     - Read the cache file created by the YELP query step to get the latitude/longitude
     - Re-construct the information and use as query key for map API
     - Pass to map API and launch google map direction on the web browser
   - Summary of data
     - # records available > 200 (estimated)
     - # records retrieved > 100 (estimated)
     - The latitude and longitude attributes in the cache created by YELP search step will used to show the direction on google map. Users are able to know how to get there and additional information on google maps.
   - Evidence of caching
     - shorturl.at/irtE0 (This one uses the cache created in Yelp query step)

C. **Database**

- Database schema
  - Basically, there are two tables, FOOD_ RESULT and RESTARUANT_ RESULT in this database.
  - Fields [type] are GenealId[int, primary key auto_increment, unique], RestaurantName [text], PhoneNumber [text], ReviewCount [real], Rating[real] , Address [text], Longitude[text], Latitude[text], UrlLink[text], Id [int]
  - The difference between two table would be the number of search results since one of the table is the search result of specifying the location.
  - The filed Id is used as "logical foreign key" – "logical primary key" within two table. In order to increase the efficiency as mentioned in the slack, I did not additionally specify new key pairs.
  - shorturl.at/dmpA9 (evidence of database schema shows here)
  - shorturl.at/iFMZ1 (evidence of the table, FOOD_RESULT )
  - shorturl.at/fHP78 (evidence of the table, RESTARUANT_ RESULT)

**D. Interaction and Presentation Plans**

Step 1:

The program would ask the user to input the interested restaurant/food and the location. (Ex: piazza annarbor)

Step 2:

The user's input would be passed into YELP API and create both cache file and SQL database. In this step, user will see their searched result with formatted text description (via command line prompts)) and compare and show their Review_count/rating in bar plot/scatter plot (via ploty).

Step 3:

The program would show the searched restaurant, and ask users which restaurant are they interested in. if they pick one of the restaurants, then it would show the related videos on YouTube. If they did not pick any, then, it will skip this step.

Step4:

The program would ask the users whether to see the relative locations of searched restaurants on the map. If yes, then it would show all the restaurants with different marks in the same static map (users are able to know the relative location of each restaurant).

Step 5:

The program would ask the users whether to launch google map direction. If yes, then after they input their desired restaurant, the program will direct them to the direction in google map. If no, then it will skip this step.

Step 6:

it will go back to step1 and keep cycling. It will not be stopped until the user enters "exit".