# Introduction

A web scrapper designed to scrap restaurants information based on a location from Grab Food Singapore Website.

Location Selected: Chong Boon Dental Surgery - Block 456 Ang Mo Kio Avenue 10, #01-1574, Singapore, 560456

# Stats

Total Restaurants Scrapped: 228

Nullable Fields: Rating, Estimated Deliver Fee, hasPromo, imageLink, chainID

QC:

* I have acknowledged the nullable fields and made the process such that any field could be nullable.

# Approach and Methodology

The basic approach is to fetch the POST requests to <https://portal.grab.com/foodweb/v2/search> which is called whenever the website wants to search in restaurants based on location.

The JSON Response Body of POST Request consists of all the information we require and we can parse it in a format that we require.

## Methodology

1. Open the link <https://food.grab.com/sg/en/> using the selenium wire’s Firefox web driver.
2. We get to the input tag element that has id “location-input” and search for “Chong Boon Dental” and this will give us the list of location options to choose from.
3. We select the location from appeared list and click on Search button.
4. Now we see that at a time we get only 32 restaurants at max in the POST request made to portal.grab.com link.
5. To load more restaurants, we need to scroll down to the bottom of the page 8 times. I selected 8 times as we need 200 restaurants minimum so 200 / 32 = 6.25, so to be safe 8 iterations.
6. Now we iterate through all the requests made by the browser and select the 8 requests we want.
7. Now these 8 requests contain nearly 200 restaurants so we use multiprocessing here to transform these restaurant’s JSON response into the structure we want and write a GZIP file out of it.

# Challenges and Solutions

1. The URL problem: Selecting address in the provided URL <https://food.grab.com/sg/en/4> was resulting in no restaurants found.
   1. Used URL: <https://food.grab.com/sg/en/>
2. The Dependencies problem: SeleniumWire hasn’t been maintained since January 2024 so issue was that last version used a dependency blinker and if we do a direct pip install, pip will install latest version of blinker and that is not compatible with selenium wire. So I had to manually downgrade to blinker version 1.7.0.
3. The Location Problem: When I was directly passing the address in search field and clicking on search button, I was still getting results of default location (Singapore). The API triggered after clicking Search button was not able to find the complete address.
   1. Breaking Address into pieces and by hit and trial, I came to a method that if just search for first 3 words of address, we can select the address from the new drop-down list that appears and using that I was able to get the results of that particular location.
4. The New Thing Problem: This was my first ever project in web scraping so it took me some time to learn it but my Python skills helped me out and I was able to finish the project in 1.5 days instead of the 3 days provided.
5. The VPN/Proxy Issue: The website was throwing 403 error multiple times due to load I guess. I used CloudFlare WARP as my DNS provider and 1 out of every 10 times I was able to connect to the website.

# Improvements and Optimizations

1. I would say that right now I am iterating through each request and then decoding them. I can use multiprocessing there also to make the process a little bit faster.
2. The sleep commands that are used are of fixed time, we could use EC to check whether the required object is already loaded or not.

# Steps to Launch in Local:

Requirements: Python > 3.6 [As the program uses f-strings in logs]

1. CD into the directory of project.
2. Create a virtual environment by using command:  
   python -m venv .pyvenv
3. To activate the virtual environment :  
   On linux use,   
   source .pyvenv/bin/activate  
   On windows use,  
   ./.pyvenv/Scripts/activate
4. Do a pip install,  
   pip install -r requirements.txt
5. Run the program:  
   python -m scrape\_data

# Comments

1. I had a lot of fun making this project as Web Scraping being a new toping for me.
2. The code for making JSON is left intentionally so as to make a human readable version of the data also.
3. I have created an out.log file also which shows the logs of entire process, might not work on some machines.