Google Maps Extractor using Python & Playwright

The design flow of the Python script involves several steps for scraping business information from Google Maps. Let's break down the design flow in detail:

1. Command-line Argument Parsing:

- The script uses the **argparse** library to parse command-line arguments.
- The two arguments are **-s** (or **--search**) for the search query and **-t** (or **--total**) for the total number of results to scrape.
- If no arguments are provided, the script prompts the user to enter a search query interactively.

2. Initialization and Setup:

- The script initializes Playwright and launches a Chromium browser.
- A new browser page is created to interact with the Google Maps website.

3. Navigating to Google Maps:

- The script navigates to the Google Maps website (**https://www.google.com/maps**) using the **goto** method.
- A timeout is set to allow the page to load fully.

4. Performing the Search:

- The script fills the search box with the specified search query using the **fill** method.
- It simulates pressing the "Enter" key to initiate the search.
- A timeout is used to wait for the search results to load.

5. Scrolling to Load More Results:

- The script simulates hovering over an element to trigger the loading of additional search results.
- A loop is used to keep scrolling until the desired number of results (**total**) is reached.
- The loop breaks if it detects that no new listings are being loaded.

6. Extracting Listings:

- The script retrieves the listings using Playwright locators.
- It creates a list of listing elements for further processing.

7. Scraping Detailed Information:

- The script iterates over each listing to click on it and extract detailed information.
- Information such as name, address, website, phone number, average rating, and reviews count is extracted using XPaths.
- The latitude and longitude coordinates are extracted from the current page URL.

8. Handling Exceptions:

- The script includes exception handling to catch any errors that may occur during scraping.
- If an exception occurs, the script prints the error message.

9. Saving Data to Excel:

- The script creates instances of **Search_Results** to store the extracted information.
- The collected data is then saved to an Excel file using the **SearchResultList** class, which provides methods for conversion to a Pandas data frame and saving to Excel.

10. Closing the Browser:

- After scraping is complete, the script closes the Chromium browser.

Note:

- The design flow ensures the automated interaction with the Google Maps website, handling dynamic loading of results, and extraction of relevant information from each listing.
- Exception handling is incorporated to address potential issues during the scraping process.
- The data is saved to an Excel file for further analysis or presentation.