from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.chrome.options import Options

from webdriver\_manager.chrome import ChromeDriverManager

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

import time

import csv

def setup\_driver():

"""Setting-up Chrome WebDriver with anti-detection options"""

chrome\_options = Options()

chrome\_options.add\_argument("--headless")

chrome\_options.add\_argument("--disable-blink-features=AutomationControlled")

chrome\_options.add\_argument("--disable-extensions")

chrome\_options.add\_experimental\_option("excludeSwitches", ["enable-automation"])

chrome\_options.add\_experimental\_option('useAutomationExtension', False)

chrome\_options.add\_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36")

service = Service(ChromeDriverManager().install())

return webdriver.Chrome(service=service, options=chrome\_options)

def scrape\_google\_results(search\_query, num\_results=10):

"""Scraping Google search results with handling"""

driver = setup\_driver()

try:

# Navigate to Google

driver.get('https://www.google.com')

time.sleep(2)

# Find and fill search input

search\_box = WebDriverWait(driver, 10).until(

EC.presence\_of\_element\_located((By.NAME, 'q'))

)

search\_box.send\_keys(search\_query)

search\_box.send\_keys(Keys.RETURN)

# Wait for results

WebDriverWait(driver, 10).until(

EC.presence\_of\_element\_located((By.CSS\_SELECTOR, 'div.g, div.yuRUbf'))

)

# Collect results

results = []

# Multiple potential selectors for maximum compatibility

title\_selectors = [

'h3',

'div.yuRUbf h3',

'div.r h3'

]

snippet\_selectors = [

'div.VwiC3b',

'div.IsZvec',

'div.s',

'div.summary'

]

for selector in title\_selectors:

try:

search\_results = driver.find\_elements(By.CSS\_SELECTOR, f'{selector} + div.g, div.g')

for result in search\_results[:num\_results]:

try:

# Extract title

title\_elem = result.find\_element(By.CSS\_SELECTOR, selector)

title = title\_elem.text

# Extract link

link\_elem = result.find\_element(By.CSS\_SELECTOR, 'a')

link = link\_elem.get\_attribute('href')

# Extract snippet (try multiple selectors)

snippet = "No snippet available"

for snippet\_selector in snippet\_selectors:

try:

snippet\_elem = result.find\_element(By.CSS\_SELECTOR, snippet\_selector)

snippet = snippet\_elem.text

break

except:

continue

results.append({

'title': title,

'link': link,

'snippet': snippet

})

except Exception as e:

print(f"Individual result extraction error: {e}")

# If results found, break the selector loop

if results:

break

except Exception as e:

print(f"Result search error with selector {selector}: {e}")

return results

except Exception as e:

print(f"Scraping error: {e}")

return []

finally:

driver.quit()

def main():

search\_term = input("Enter your query : ")

results = scrape\_google\_results(search\_term)

if results:

print("Here's your search result : ")

for result in results:

print(f"Title: {result['title']}")

print(f"Link: {result['link']}")

print(f"Snippet: {result['snippet']}\n")

# Save to CSV

with open('scraped\_google\_search\_results.csv', 'w', newline='', encoding='utf-8') as f:

writer = csv.DictWriter(f, fieldnames=['title', 'link', 'snippet'])

writer.writeheader()

writer.writerows(results)

print(f"Saved {len(results)} results to google\_search\_results.csv")

else:

print("No results found.")

if \_\_name\_\_ == "\_\_main\_\_":

main()