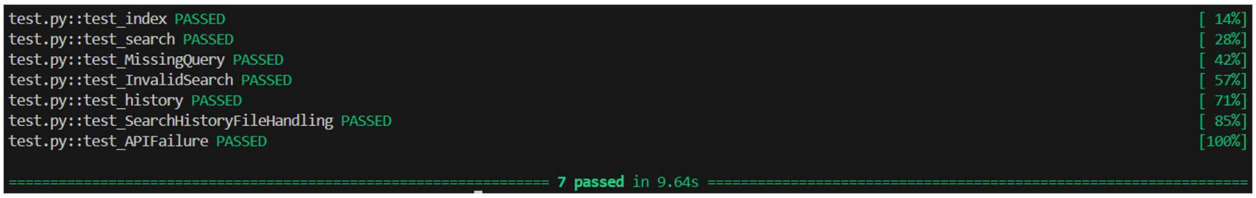
COSC 4P02 – Aggregator Test Document:

Completed Automated Tests:



Test Execution Commands:

python -m pytest test.py -v

python -m pytest test.py -s -v (this will display the messages and content printed in the tests when run)

python -m pytest test.py -k "test_index" -s -v

python -m pytest test.py -k "test_search" -s -v

python -m pytest test.py -k "test_MissingQuery" -s -v

python -m pytest test.py -k "test_InvalidSearch" -s -v

python -m pytest test.py -k "test_history" -s -v

python -m pytest test.py -k "test_SearchHistoryFileHandling" -s -v

python -m pytest test.py -k "test_APIFailure" -s -v

Method Test Cases and Descriptions:

test_index:

Test Case 1: Testing the index of the aggregator. This will confirm that the index page is reachable and correct. This method will test the index page of the Flask app. It sends a GET request to the root URL ('/') and checks if the response status code is 200 (OK) and if the response data contains the text "Keyword Search Aggregator".

Expected Result: Pass. The page should load successfully and display the "Keyword Search Aggregator" text.

test_search:

Test Case 2: Testing the search functionality of the `/search` route with a query parameter. This method verifies that the search functionality works when the user submits a query. The test sends a GET request to the `/search` endpoint with the query parameter, checks that the status code is 200 (OK), and prints the first 10 search result URLs from both Google

Search and Google News. It also ensures that the response contains the expected "organic_results" and "news_results" for Google Search and Google News respectively.

Expected Result: Pass. The search functionality should return valid search results, and the results should contain the appropriate URLs.

test_MissingQuery:

Test Case 3: Testing the aggregator when the query parameter is missing from the `/search` route. This method verifies that the `/search` route returns a 400-status code and the appropriate error message when no query parameter is provided. The test sends a GET request to the `/search` endpoint without the query parameter and checks that the response status code is 400 (Bad Request). It also ensures that the response contains the correct error message indicating that the query parameter 'q' is missing.

Expected Result: Pass. The route should return a 400-status code and an error message stating that the 'q' parameter is missing.

test_InvalidSearch:

Test Case 4: Testing the `/search` route with an invalid keyword ("!!!"). This method verifies that when an invalid keyword "!!!" is provided, the API returns no search results. It checks that both the Google Search and Google News sections are empty in the response.

Expected Result: Pass. The search results should be empty for both Google Search and Google News.

test_history:

Test Case 5: Testing the history functionality of the `/history` route to retrieve stored search history. This method verifies that the history functionality works as expected. It sends a GET request to the `/history` route, checks that the response status code is 200 (OK), and prints the search history entries along with the associated URLs and sources. It also ensures that the response is a list, even if it is empty.

Expected Result: Pass. The history functionality should return valid search history entries, and the response should be a list of records with keywords and results.

test_SearchHistoryFileHandling:

Test Case 6: Testing the creation and updating of “search_history.json” file. This test ensures that when a search is performed and “search_history.json” does not exist, it is created. Additionally, it checks that the file is updated properly when new searches are performed.

Expected Result: Pass. The file should be created if it doesn't exist and updated with the new search history when new searches are added.

test_APIFailure:

Test Case 7: Testing the `/search` route when the SerpAPI fails to respond. This method verifies that when the SerpAPI service fails, the `/search` endpoint handles the failure gracefully by returning a 500-status code and an appropriate error message. It uses a mock request to simulate a failure when contacting the SerpAPI service.

Expected Result: Pass. The `/search` route should return a 500-status code and an error message indicating the failure to contact the SerpAPI or another internal error.