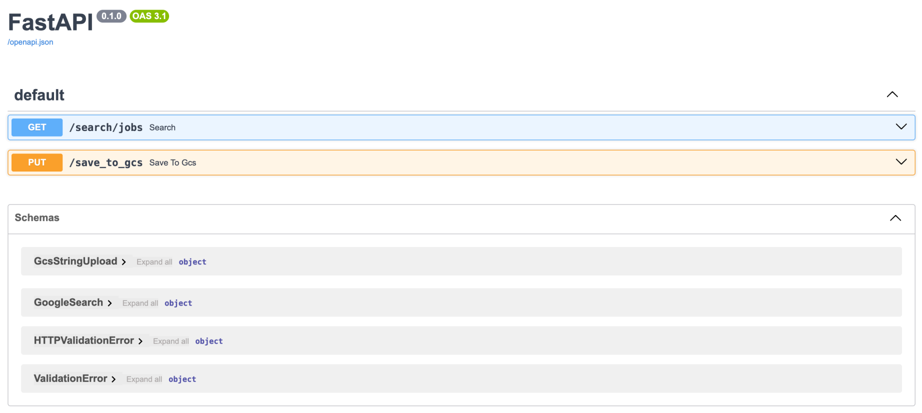
Individual Assignment 2

Goal

1. Build a FastAPI server that:
- Queries job postings from Google Custom Search API.
- Saves results to Google Cloud Storage.



FastAPI 0.1.0 OAS 3.1
/openapi.json

default

GET `/search/jobs` Search

PUT `/save_to_gcs` Save To Gcs

Schemas

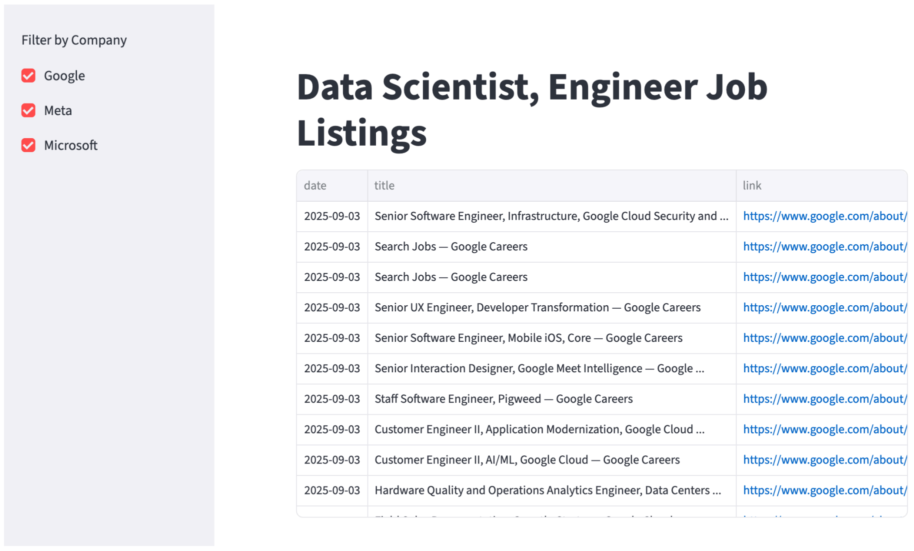
GcsStringUpload › Expand all object

GoogleSearch › Expand all object

HTTPValidationError › Expand all object

ValidationError › Expand all object

2. Extend HW 1 to build a Streamlit app that:
- Retrieves stored job results from GCS.
- Displays them interactively in a table.



Filter by Company

- [x] Google
- [x] Meta
- [x] Microsoft

Data Scientist, Engineer Job Listings

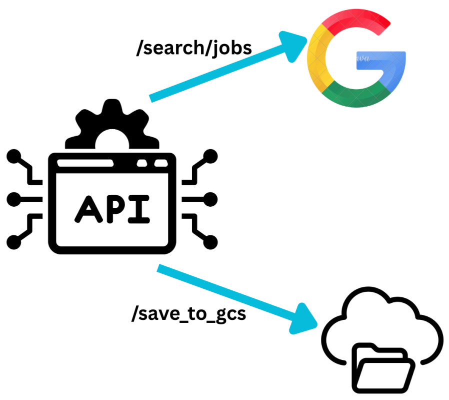
date	title	link
2025-09-03	Senior Software Engineer, Infrastructure, Google Cloud Security and ...	https://www.google.com/about/
2025-09-03	Search Jobs — Google Careers	https://www.google.com/about/
2025-09-03	Search Jobs — Google Careers	https://www.google.com/about/
2025-09-03	Senior UX Engineer, Developer Transformation — Google Careers	https://www.google.com/about/
2025-09-03	Senior Software Engineer, Mobile iOS, Core — Google Careers	https://www.google.com/about/
2025-09-03	Senior Interaction Designer, Google Meet Intelligence — Google ...	https://www.google.com/about/
2025-09-03	Staff Software Engineer, Pigweed — Google Careers	https://www.google.com/about/
2025-09-03	Customer Engineer II, Application Modernization, Google Cloud ...	https://www.google.com/about/
2025-09-03	Customer Engineer II, AI/ML, Google Cloud — Google Careers	https://www.google.com/about/
2025-09-03	Hardware Quality and Operations Analytics Engineer, Data Centers ...	https://www.google.com/about/

You can work on the streamlit application independently from the fastAPI server using the provided **`Example_Data`**

Details & Requirements

---

1. FastAPI Server



Implement the following endpoints in a file called extract_save_data.py.
You can run the FastAPI application by `$fastapi dev extract_save_data.py`
See user_definition.py and update your .env file to store sensitive information.
A and B are working, if you run `$python call_fast_api.py` it should print the following lines.

search_response status: 200
upload_response status: 200

A. POST /search/jobs (1.5 PT)
Refer to
https://developers.google.com/custom-search/v1/overview **(Make sure to create API_KEY and SEARCH_ENGINE_ID and add to .env)**
https://developers.google.com/custom-search/v1/reference/rest/v1/Search

Note :
I provided '**`call_fast_api.py`**' that you can test '**`/search/jobs`**' and '**`/save_to_gcs`**'.
- If you run this script, it will run a google custom search, clean and save it to a GCS bucket, if your code is correctly done.
- You can also refer example .json files in the provided '**`Example_Data`**' to check the expected format

Behavior:
1. Call the Google Custom Search API.
   - If the search returns more than 100 matches, it should limit the matches to 100.
   - Construct a search query using **`job_title`** and **`company_dictionary`**.
   - Parameters should be properly assigned including **`key`**, **`cx`**, **`query`**, and **`dateRestrict`**, etc.
     - This is mentioned in the API documentation.

2. Parse the response into a list of job postings with **`title, link, snippet`**, and **`date`**.
   - `title, link`, and `snippet` are from **`search_results`**' "**`items`**".
   - **`date`** is based on the snippet's "xx days ago". You can use the current date and xx days to calculate it.

Input: JSON body (Use GoogleSearch model)

Output: Return a dictionary of

```
 {"company_dict": company_dict used for search,
  "job_title":  job_title used for search,
  "results": a list of dictionaries with title, link, snippet, and date. }
```

Ex.

```
{
  "company_dict": {...},
  "job_title": "Data Scientist",
  "results": [
    {"title": "...", "link": "...", "snippet": "...", "date": "2025-09-03"},
    ...
  ]
}
```

B. PUT /save_to_gcs (0.8 PT)

Behavior:

1. Save the provided string to the specified GCS bucket.

Input: JSON body (Use **`GcsStringUpload`** model in the code)

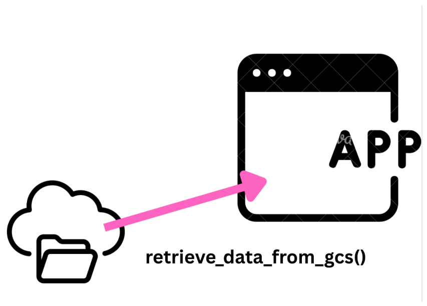
Output: Return '`{"message": f"file {gcs_upload_param.file_name} has been uploaded  to {gcs_upload_param.bucket_name} successfully."}`'

Ex.

```
{"message": "file jobs_search/2025-09-03.json has been uploaded to msds692
successfully."}
```

---

2. Streamlit Web Application



Extend hw1.py.

You can run the Streamlit application by `$streamlit run main.py`

A. `retrieve_data_from_gcs(service_account_key: str,  project_id: str, bucket_name: str, file_name_prefix: str )` (1 PT)

Behavior:

Retrieve file contents from all files starting with "`file_name_prefix`" in "`bucket_name`" and returns a dictionary including "`results`", "`job_titles`", and "`company_dict`"

Input Parameters:

- `service_account_key` (str) : path of service account key file(.json)
- `project_id` (str) : GCP Project ID where bucket is located
- `bucket_name` (str) : bucket name
- `file_name_prefix` (str) : prefix of files to retrieve data. (Ex.”job_search/“)

Output: Return

```
{"results": a list including "results" from all the files starting
with file_name_prefix,
  "job_titles": a list including unique "job_title"s from all the
files starting with file_name_prefix,
  "company_dict": a dictionary including all "company_dict"s from all
the files starting with file_name_prefix}
```

B. main (0.5 PT)
Behavior:
Mostly the same as hw1, but make sure of the following differences.

- Title should be comma-separated strings of job titles in ascending order.
- Company list on the sidebar should include unique names in ascending order.
- The dataframe should only include unique values.

Note:
If you want to try to build this first, you can place the provided .json files in the provided.
I recommend you to try with one file, and then add the other to see whether it displays title correctly (there are two different titles), and dynamically add companies from the data file on the checkbox options.

---

3. Code Quality (0.2 PT)

- Do not hardcode variables. Use variables from `user_definition.py`.
- Your code must pass style checks with fewer than 5 PEP8 issues using `$ pycodestyle hw2.py`

Please see the provided URLs as a reference.
Streamlit : https://dwoodbridge-hw2-api-477009951698.europe-west1.run.app/
FastAPI: https://dwoodbridge-hw2-fastapi-477009951698.europe-west1.run.app/docs

Make sure it passes the provided pytest.