Adecco Software Engineer Test

Prerequisites:

1. Visual Studio/VS Code/Jetbrains Rider IDE
2. React/Node js/typescript/js/jquery… (The tools the dev need could need to perform the task)
3. Own a github account

Considerations

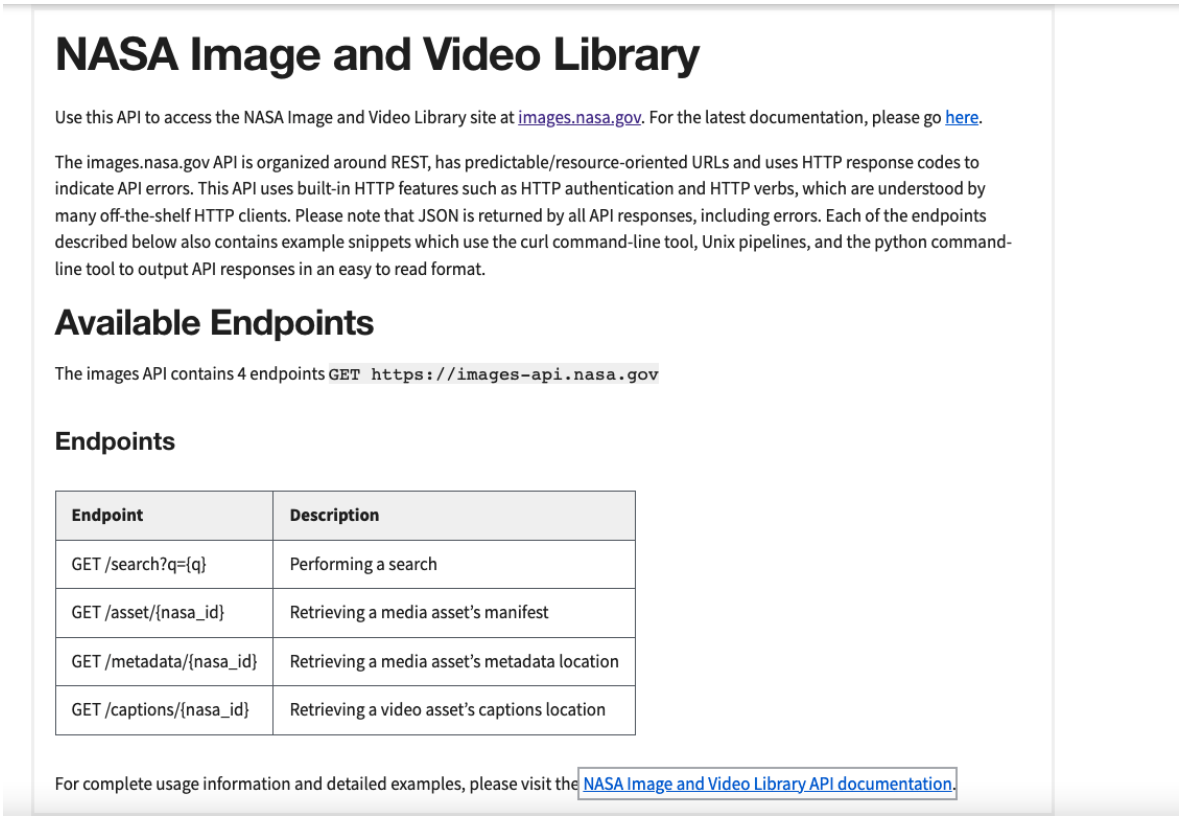
- Recommended time to be spent: 8-10h (with a break). But we are not so strict with the time spent.
- If there are things that can be improved please name them in a text file including them.
- The results have to be sent at the end of the day, uploaded in a github account.

We need an application that show a list of NASA pics. The documentation can be found here:

https://api.nasa.gov/

Steps for accessing the API:

NASA Image and Video Library: API to access the NASA Image and Video Library site at images.nasa.gov



NASA Image and Video Library

Use this API to access the NASA Image and Video Library site at images.nasa.gov. For the latest documentation, please go here.

The images.nasa.gov API is organized around REST, has predictable/resource-oriented URLs and uses HTTP response codes to indicate API errors. This API uses built-in HTTP features such as HTTP authentication and HTTP verbs, which are understood by many off-the-shelf HTTP clients. Please note that JSON is returned by all API responses, including errors. Each of the endpoints described below also contains example snippets which use the curl command-line tool, Unix pipelines, and the python command-line tool to output API responses in an easy to read format.

Available Endpoints

The images API contains 4 endpoints `GET https://images-api.nasa.gov`

Endpoints

Endpoint	Description
GET /search?q={q}	Performing a search
GET /asset/{nasa_id}	Retrieving a media asset's manifest
GET /metadata/{nasa_id}	Retrieving a media asset's metadata location
GET /captions/{nasa_id}	Retrieving a video asset's captions location

For complete usage information and detailed examples, please visit the NASA Image and Video Library API documentation.

In that section you will find the necessary documentation to access the api

Minimum Requirements:

- Frontend
    - We need to show above some filters (at least):
        - Search Query (e.g: Apollo 11, stars…)
        - Year start date
        - Year end date
        - Media_type (image)
    - Base on the filtering a search is performed and shows a list with pictures
    - We can show up to 10 (or more, it's not a problem)
    - Each entry (either a picture or a text) should be clickable and navigates to the photo and with the description and the details that are provided from the API (only the relevant ones are needed)
    - The nicer and organized it looks the better **but making it work it's the most important thing so leave this improvement for the end.**
- The backend should be a Web API, well-structured and organized. It should be kind of a intermediate layer to call the NASA API.
- Using GitHub with multiple check-ins.
- Well structured and clean project.
- **Clear understanding of the patterns technologies used.**

Will be great if the project includes

- **Caching (are the results collected the same based on the filter values?)**
- **Unit Tests** (indications of what could be tested is good if there is no time)
- **Logging, error handling, organized project structure, any design pattern, settings configuration, authorization…**
- **Postman requests**

UI improvements (less important as mentioned but Good it they are nice)

- Adding pagination or loading more on scroll is optional
- Adding filtering by name in the main view so only the matching results are shown.
- Improvements that you could have added to the project