Project's report

I. Introduction:

A simple website querying inputed keyword(s) using Google searching engine through SerpAPI, returning the query's organic results with corresponding links and snippets. The server is hosted on Render and is called directly on frontend web page.

II. Front-end:

1. Elements:

There is an input box for keywords, a div tag for presenting fetched results and a button to download the json formatted strutured result.

```
<body>
    <input type="text" id="input"
        placeholder="Search something..."
        onchange="handleOnchangeInput(this.value.trim())"/>
    <div id="output"></div>
    <button id="DownloadButton" onclick="downloadJSON()" disabled>Download JSON</button>
```

2. Scripts:

```
<script>
    let organicResults = [];

    async function handleOnchangeInput(keyWord) {···

    function downloadJSON() {···
</script>
```

There is an array to stored organic results as dictionaries, a function fetching data for keyword(s) from server and a function to download fetched results for download button.

- **Taking data function:** as the only API for frontend, I put it into index.html for convenience.

```
async function handleOnchangeInput(keyWord) {
    const downloadBtn = document.getElementById("DownloadButton");
    const output = document.getElementById("output");
    downloadBtn.disabled = true;
    output.innerHTML = '';
    if (!keyWord) return;

    try {···
        output.innerHTML = error;
    }
}
```

The function identify the elements to be interacted while also handling input.

Then make an effort to request data from the server. If there any problems occurs, an errors will be displayed on the output div tag. Otherwise, download button will abled, div tags with link and snippet of the results will be added to output div tag.

The link are handled to be opened in another tab and prevent a type of phising attack called tabnabbing. It's a type of attack that the attacker navigate the original tab of opened linked tab, which has been left unattended for a period of time, with a fake login page, tricking the users to enter their credentials.

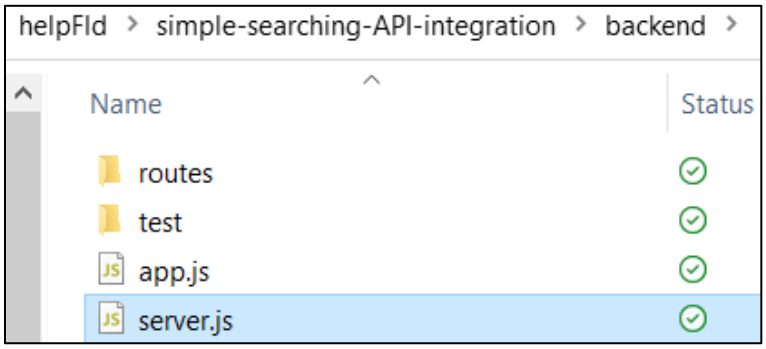
```
try {
    const response = await fetch(`https://simple-searching-api-i
    const data = await response.json();

    if (data.error) throw data.error;
    organicResults = (data.organic_results || []).map(item => ({
                        title: item.title,
                        link: item.link,
                        snippet: item.snippet
                    }))
    if (organicResults.length === 0) {
        output.innerHTML = "No results found";
        return;
    }
    downloadBtn.disabled = false;

    organicResults.forEach(item => {
        const a = document.createElement("a");
        a.innerHTML = `&bull; ${item.title}`;
        a.href = item.link;
        a.target = "_blank"; // this make the link opened in a n
        a.rel = "noopener noreferrer"; // rel sets the relations
        // set it like above prevents a type of phising called t

        const p = document.createElement("p");
        p.innerHTML = item.snippet;

        const divRes = document.createElement("div");
        divRes.appendChild(a); divRes.appendChild(p);
        output.appendChild(divRes);
    })
} catch (error) {
    output.innerHTML = error;
}
```

- **Download data function:** when the download button is clicked, a link element will be created and clicked to download a stringify dictionary result into a json file.

```
function downloadJSON() {
    const blob = new Blob([JSON.stringify(organicResults, null, 2)],
                          {type: 'application/json'});
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = 'search_results.json';
    a.click();
}
```

III. Back-end:

Contains an **app.js** as the main source code using APIs from **routes**, an **server.js** using app from **app.js** and declare port to run on, **routes folder** contains all the API source code and **test folder** containing .test.js files for testing purpose using jest and supertest module.



- Routes: only has a js file for search API at the moment, receiving keyword as input and response with dictionary containing either the data of the API or an error and corresponding status code.

```
module.exports = async (req, res) => {
    const keyWord = req.query.q;
    if (!keyWord) return res.status(400).json({error: 'Missing key word'});

    try {
        const url = `https://serpapi.com/search.json?engine=google&q=${keyW
        // const url = `https://cors-anywhere.herokuapp.com/google.com/sear
        // not stable, there is a rate limit, easily blocked
        const response = await fetch(url);
        const data = await response.json();

        res.status(200).json(data);
    } catch (error) {
        res.status(500).json({error: 'Failed to fetch from SerpAPI'});
    }
}
```

- Test: containing a search.test.js containing all the cases for searching test including normal case and empty input case, there is only two cases for now. Tests are designed with corresponding description and callback function that uses expect to decide if the test is pass or not.

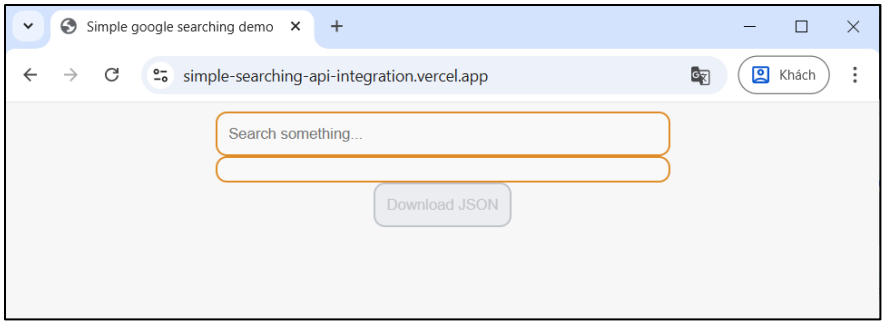
```
const request = require('supertest');
const app = require('../app');

test("Return normal result, no errors: check if the result has s
    const response = await request(app).get('/search?q=hanoi');
    const results = response.body.organic_results;

    expect(response.status).toBe(200);
    if (results.length > 0) {
        expect(results[0]).toHaveProperty('title');
        expect(results[0]).toHaveProperty('link');
        expect(results[0]).toHaveProperty('snippet');
    }
});

test("Empty input, return error status 400 with error element.",
    const result = await request(app).get('/search?q=');

    expect(result.status).toBe(400);
    expect(result).toHaveProperty('error');
});
```

- App.js: importing source code from routes folder to use for API's path declarations.
- Server.js: using app and declaring PORT to work on. The port number is required for the Render platform to be able to run the server source code.

```
const express = require('express');
const cors = require('cors');
const searchRoute = require('./routes/search');
const app = express();
app.use(cors());

app.get('/search', searchRoute);

module.exports = app;
```

```
const app = require('./app');
const PORT = 4000;

app.listen(PORT, () => {console.log(`Server is running on ${PORT}`);});
```

IV. Result:



habanera

• Carmen - Habanera (Bizet; Anna Caterina Antonacci, The ...

An opera about Spain, written in French. Makes perfect sense. Still, love is a rebellious bird. Beautiful.

• Habanera (aria)

Habanera is the popular name for "L'amour est un oiseau rebelle an aria from Georges Bizet's 1875 opéra comique Carmen. It is the entrance aria of the title ...

• The Global Habanera

And finally, the contredanse and habanera are social dances traditionally performed by couples facing each other in a long line or in a square formation.

• Habanera

Music · Habanera or contradanza, a style of Cuban popular dance music of the 19th century · Habanera, a work for violin and piano by Pablo de Sarasate, part of ...

• Habanera - Musical Atlas of Cuba

The habanera's distinguishing musical feature is its short, repeating 2/4 rhythmic figure in the bass line.

• Habanera from Carmen - Bizet (Cello & Piano)

If technology has a bright side, it's that musicians like this are able to deliver this loveliness into my home, at no charge. Thank you!

• Maria Callas – Habanera from Bizet's Carmen ("L'amour est ...

Carmen's vivacious Habanera plays out in illustration to the sound of Maria Callas's voice in this captivating video by Matteo Cozzo, ...

• Maria Callas Live: Bizet's Carmen Habanera, Hamburg 1962

http://www.maria-callas.com Callas's only operatic appearances in Germany were Lucia di Lammermoor, with

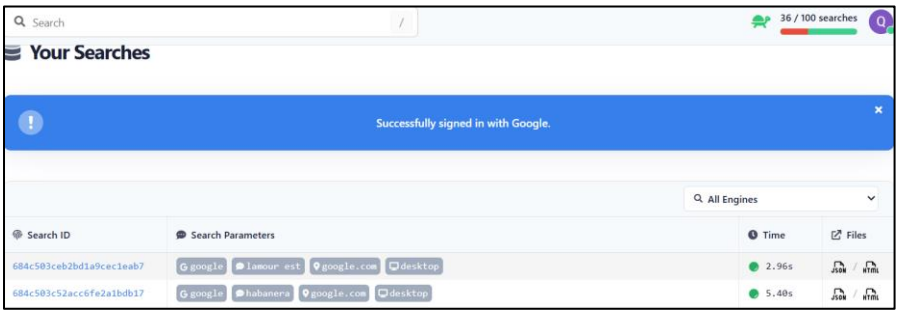
• Maria Callas Live: Bizet's Carmen Habanera, Hamburg 1962

http://www.maria-callas.com Callas's only operatic appearances in Germany were Lucia di Lammermoor, with Karajan conducting, ...

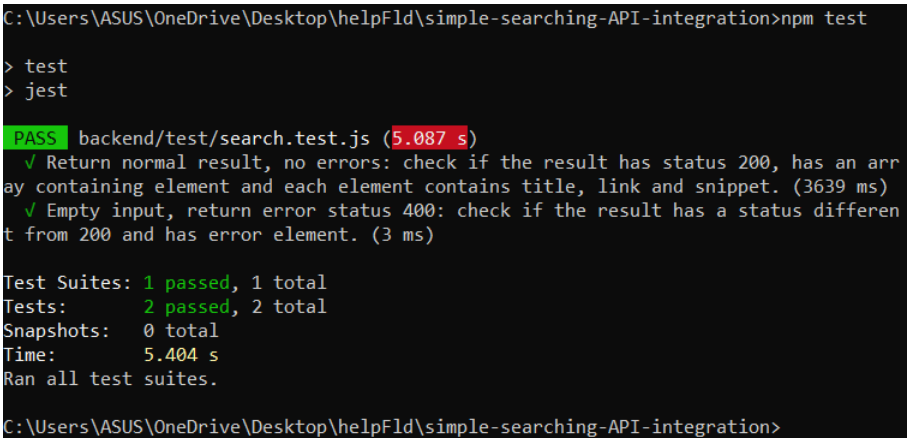
• Maria Callas - Habanera (Lyric Video)

Maria Callas performs the "Habanera" (L'amour est un oiseau rebelle) from Bizet's Opera Carmen. Feel free to sing along!

Download JSON



V. Test:


C:\Users\ASUS\OneDrive\Desktop\helpFld\simple-searching-API-integration>npm test
> test
> jest
PASS backend/test/search.test.js (5.087 s)
√ Return normal result, no errors: check if the result has status 200, has an arr
ay containing element and each element contains title, link and snippet. (3639 ms)
√ Empty input, return error status 400: check if the result has a status differen
t from 200 and has error element. (3 ms)
Test Suites: 1 passed, 1 total
Tests: 2 passed, 2 total
Snapshots: 0 total
Time: 5.404 s
Ran all test suites.
C:\Users\ASUS\OneDrive\Desktop\helpFld\simple-searching-API-integration>