IndiaWebScraperDocumentation

V1.0

**Table of Contents:**

|  |  |
| --- | --- |
| Overview |  |
| Key Features |  |
| Code Analysis |  |
| Resources/Research |  |

**Overview of the Web Scraper:**

This web scraper is a Python tool built to pull job listing data from websites. It goes through job pages, grabs details like job titles, companies, locations, descriptions, and URLs.

**How It Works**

* **Collects Data**: The scraper automatically goes through the web pages and pulls out the important info for each job.
* **Organizes It**: All the data gets neatly structured into a list, with each job's details grouped together.

**Why CSV?**

The scraper saves everything into a **CSV file** because:

* **It's Easy to Use**: CSV files are simple and work across different platforms.
* **User-Friendly**: You can open the CSV in Excel or similar programs to view the data without needing any special tools.

**Key Features:**

* **Web Scraper built to find details on job search information:**
* **Job title**
* **Company**
* **Location**
* **Date**
* **Skills**
* **Experience**
* **Qualifications**
* **Remote/On Site/Contractor**
* **URL**
* **Export to CSV for easy readability**

**Code Analysis:**

Code Snippet Edited

import requests

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query":"cybersecurity jobs on glassdoor","page":"1","num\_pages":"1","date\_posted":"all"}

headers = {

    "x-rapidapi-key": "insert key",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

response = requests.get(url, headers=headers, params=querystring)

print(response.json())

**INITIAL CODE:**

**Taken from Jsearch API to scrape for cybersecurity jobs on the website "glassdoor". VS Code was used to run all scripts.**

**Summary of the Code Flow:**

1. **Imports the requests library** to make HTTP requests.
2. **Defines the API endpoint** (url) you want to query.
3. **Sets up the query parameters** (querystring) that specify the search details (e.g., job type, page number).
4. **Adds the necessary headers** (headers), including your API key for authentication.
5. **Sends the HTTP GET request** to the API using requests.get() with the URL, headers, and parameters.
6. **Prints the JSON response** from the API in a readable format using .json().

Json/ CSV File Breakdown

**Code ouput was running in terminal only**:

**Modify the Script to Save the JSON Response to a File**

You can save the response JSON into a file (e.g., cybersecurity\_jobs.json) in the project directory. Here's the updated version of the script:

import requests

import json

# Define the API endpoint and query parameters

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query":"cybersecurity jobs on glassdoor","page":"1","num\_pages":"1","date\_posted":"all"}

# Add your RapidAPI headers

headers = {

    "x-rapidapi-key": "insert key",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

response = requests.get(url, headers=headers, params=querystring)

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Save the JSON response to a file

    with open("cybersecurity\_jobs.json", "w") as json\_file:

        json.dump(data, json\_file, indent=4)  # Pretty print with indentation for readability

    print("Data saved to 'cybersecurity\_jobs.json'")

else:

    print(f"Failed to retrieve data. Status code: {response.status\_code}")

**WHAT CHANGED:**

1. **Importing the json library**: To save the response in JSON format.
2. **json.dump()**: This function is used to write the JSON response into a file.
3. **File Creation**: The JSON response is saved to a file named cybersecurity\_jobs.json in the current directory.

**WHAT WILL HAPPEN:**

* When you run the script, it will fetch the job data from RapidAPI.
* Instead of just printing the data to the terminal, it will create a file cybersecurity\_jobs.json in the project directory. This file should appear in the left panel in VS Code (the file explorer).

CSV Breakdown/Execution

**To also export the data to a CSV file, you can modify the script further by using Python's csv module to write the relevant job information into a CSV format.**

**Here's how to modify the script to export the data to a CSV file:**

**Updated Script for CSV Export**

import requests

import json

import csv

# Define the API endpoint and query parameters

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query":"cybersecurity jobs on glassdoor","page":"1","num\_pages":"1","date\_posted":"all"}

# Add your RapidAPI headers

headers = {

    "x-rapidapi-key": "insert key",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

response = requests.get(url, headers=headers, params=querystring)

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Save the JSON response to a file

    with open("cybersecurity\_jobs.json", "w") as json\_file:

        json.dump(data, json\_file, indent=4)  # Pretty print with indentation for readability

    print("Data saved to 'cybersecurity\_jobs.json'")

    # Export relevant job data to CSV

    job\_listings = data.get("data", [])  # Assuming the job listings are stored in the "data" key

    if job\_listings:

        # Define the CSV file and headers

        with open("cybersecurity\_jobs.csv", "w", newline="", encoding="utf-8") as csv\_file:

            csv\_writer = csv.writer(csv\_file)

            # Write the CSV header

            csv\_writer.writerow(["Job Title", "Company", "Location", "Date Posted", "Job Description", "URL"])

            # Write the job data to the CSV file

            for job in job\_listings:

                job\_title = job.get("title", "N/A")

                company = job.get("company\_name", "N/A")

                location = job.get("location", "N/A")

                date\_posted = job.get("posted\_at", "N/A")

                description = job.get("description", "N/A")

                url = job.get("url", "N/A")

                # Write the row to CSV

                csv\_writer.writerow([job\_title, company, location, date\_posted, description, url])

        print("Data exported to 'cybersecurity\_jobs.csv'")

    else:

        print("No job listings found in the response.")

else:

    print(f"Failed to retrieve data. Status code: {response.status\_code}")

**Key Changes:**

* 1. **Importing the csv module**: This will allow us to write the job data to a CSV file.
  2. **Fetching the relevant data**: We're pulling specific fields like the job title, company name, location, date posted, description, and URL from the JSON response.
  3. **Writing to CSV**: We're opening a CSV file (cybersecurity\_jobs.csv) and writing the job data into it row by row, with headers.

**CSV Fields:**

* 1. **Job Title**: The title of the job.
  2. **Company**: The name of the company offering the job.
  3. **Location**: Where the job is located.
  4. **Date Posted**: When the job was posted.
  5. **Job Description**: A brief description of the job.
  6. **URL**: The link to the job listing.

**What Will Happen:**

* 1. After the API request is completed, the script will now:
     1. Save the JSON response to cybersecurity\_jobs.json.
     2. Create and write the job listings to cybersecurity\_jobs.csv.

Both files should appear in your VS Code file explorer, and you’ll have the job details in a CSV file.

**\*\*\*At this point, the CSV file in VS CODE was returning missing values, Troubleshooting for that below:**

import requests

import json

import csv

# API request setup

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query": "cybersecurity jobs on glassdoor", "page": "1", "num\_pages": "1", "date\_posted": "all"}

headers = {

    "x-rapidapi-key": "insert key",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

response = requests.get(url, headers=headers, params=querystring)

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Pretty print the JSON response to inspect the structure

    print(json.dumps(data, indent=4))  # Inspect the structure of the response

    # Save the JSON response to a file (optional, for later review)

    with open("cybersecurity\_jobs.json", "w") as json\_file:

        json.dump(data, json\_file, indent=4)

    # Open or create a CSV file to write job data

    with open("cybersecurity\_jobs.csv", mode="w", newline="", encoding="utf-8") as file:

        writer = csv.writer(file)

        # Write the headers

        writer.writerow(["Job Title", "Company", "Location", "Date Posted", "Job Description", "URL"])

        # Assuming the jobs data is in the 'data' key

        if 'data' in data:

            jobs = data['data']

            # Iterate over the job results

            for job in jobs:

                # Extract the relevant information from each job

                title = job.get('job\_title', 'N/A')

                company = job.get('employer\_name', 'N/A')

                location = job.get('location', 'N/A')  # You need to inspect and adjust this key

                date\_posted = job.get('date\_posted', 'N/A')  # Inspect and adjust this key

                job\_description = job.get('description', 'N/A')  # Inspect and adjust this key

                job\_url = job.get('job\_apply\_link', 'N/A')

                # Write the row to the CSV

                writer.writerow([title, company, location, date\_posted, job\_description, job\_url])

        else:

            print("No job data found in the response.")

else:

    print(f"Failed to retrieve data: {response.status\_code}")

**Next Steps:**

* 1. **Run the script and inspect the print(json.dumps(data, indent=4)) output**.
  2. **Check the actual keys used for fields like location, date\_posted, and description.**
  3. **Replace job.get('location', 'N/A') with the correct key you find in the JSON.**

**Once the structure is confirmed, the N/A values should be replaced with the correct data from the API response.**

**NOTE**: At this point, after modifying a few more times I still kept encountering a few errors/mix-ups:

* 1. The CSV file ended up showing N/A in the location and job description fields in excel, even though the Json file extracted the information.

* 1. The headers in excel did not show up all the necessary criteria that was asked for in the assignment

* 1. Made a mistake editing the code, ended up getting confusing data outputs so had to modify it to re-run the desired information.

**TROUBLESHOOTING:** These are the changes I made to get the final code script:

* 1. ADJUST HEADERS: Added in the additional header fields in the "writer.writerow" section in the code so it shows exactly what I want in excel file (removed job description after inspecting the Json because it was outputting unfinished/unclear sentences, and missing information in other areas).

* 1. ITERATE OVER THE JOB RESULTS: Modified to include city, state, country for the location

* 1. EXTRACT: Skills, responsibilities, experience, qualifications, remote, on site or contactor( previous CSV did not include this)

 Production-Final Scraper

**This currently fulfils the criteria for: job title, company, city, state, country, date posted, skills, experience and responsibilities.**

**\*\*\*Moved code from testbackup1.py to prod.py and this outputs to cybersecurity\_jobs\_101\_v3.csv**

**Next Steps:**

* **Separate experience and responsibilities into two separate columns (currently the 'experience' data is lumped in with the responsibilties section)**
* **Remove odd characters from the skills and qualifications columns**
* **Get the Remote, On Site, Contractor fields to show the accurate information (currently showing yes/no only even though the data states which posting is full time, remote etc)**

import requests

import json

import csv

from datetime import datetime

# Helper function to extract skills and qualifications

def extract\_skills(job\_data):

    potential\_keys = ['job\_required\_skills', 'skills', 'certifications', 'qualifications', 'job\_description']

    collected\_skills = []

    for key in potential\_keys:

        content = job\_data.get(key)

        if content:

            if isinstance(content, list):

                collected\_skills.extend(content)  # Extend list if content is a list

            elif 'qualification' in key.lower() or 'skill' in key.lower() or 'description' in key.lower():

                lines = content.split('.')

                for line in lines:

                    if "qualifications" in line.lower() or "skills" in line.lower():

                        collected\_skills.append(line.strip())

    return ', '.join(collected\_skills) if collected\_skills else 'N/A'

# API request setup

url = "<https://jsearch.p.rapidapi.com/search>"

querystring = {"query": "cybersecurity jobs on glassdoor", "page": "1", "num\_pages": "1", "date\_posted": "all"}

headers = {

    "x-rapidapi-key": "insert key",

    "x-rapidapi-host": "jsearch.p.rapidapi.com"

}

# Make the request to the API

response = requests.get(url, headers=headers, params=querystring)

# Check if the request was successful (status code 200)

if response.status\_code == 200:

    # Parse the JSON response

    data = response.json()

    # Open or create a CSV file to write job data

    with open("C:/Users/garne/Documents/cybersecurity\_jobs\_101\_v3.csv", mode="w", newline="", encoding="utf-8") as file:

        writer = csv.writer(file)

        # Write the headers

        writer.writerow(["Job Title", "Company", "City", "State", "Country", "Date Posted", "Skills", "Responsibilities", "Experience", "Qualifications", "On Site", "Remote", "Contractor", "URL"])

        # Assuming the jobs data is in the 'data' key

        if 'data' in data:

            jobs = data['data']

            # Iterate over the job results

            for job in jobs:

                # Extract the relevant information from each job

                title = job.get('job\_title', 'N/A')

                company = job.get('employer\_name', 'N/A')

                city = job.get('job\_city', 'N/A')

                state = job.get('job\_state', 'N/A')

                country = job.get('job\_country', 'N/A')

                timestamp = job.get('job\_posted\_at\_timestamp', None)

                date\_posted = datetime.utcfromtimestamp(timestamp).strftime('%Y-%m-%d') if timestamp else 'N/A'

                # Extract Skills with enhanced details

                skills = extract\_skills(job)

                # Responsibilities (inferred from job description)

                responsibilities = job.get('job\_description', 'N/A')

                # Experience (nested inside 'job\_required\_experience')

                experience = job.get('job\_required\_experience', {}).get('experience', 'N/A')

                # Qualifications (extracted and processed separately)

                qualifications = extract\_skills(job)  # Reusing the function to get detailed qualifications too

                # On Site / Remote / Contractor status

                on\_site = 'Yes' if job.get('job\_is\_on\_site', False) else 'No'

                remote = 'Yes' if job.get('job\_is\_remote', False) else 'No'

                contractor = 'Yes' if job.get('job\_is\_contract', False) else 'No'

                # Extract the job application link

                job\_url = job.get('job\_apply\_link', 'N/A')

                # Write the row to the CSV

                writer.writerow([title, company, city, state, country, date\_posted, skills, responsibilities, experience, qualifications, on\_site, remote, contractor, job\_url])

        else:

            print("No job data found in the response.")

else:

    print(f"Failed to retrieve data: {response.status\_code}")

**Resources/Research:**

**VS Code (For creating/running the scripts)**

**CoPilot**

**Claude**

**ChatGPT**

**YouTube (Visual learning for navigating VS code)**

[**Glassdoor | Job Search & Career Community**](https://www.glassdoor.com/index.htm)

[**Welcome to Rapid’s new Experience! 🚀 (rapidapi.com)**](https://rapidapi.com/hub)

[**GitHub**](https://github.com/)