PROJECT REPORT

ON

SIMPLE WEB SCRAPER

Submitted by

Anuraj Parameswaran

TABLE OF CONTENTS

1. ABSTRACT

The Simple Web Scraper is a Python-based application designed to extract remote job listings from RemoteOK.com, a popular platform for remote work opportunities. This tool automates the process of job hunting by scraping relevant job postings based on user-specified keywords such as programming languages, technologies, or job roles. The application processes the extracted data and saves it in a structured CSV format for further analysis.

The scraper utilizes modern web scraping techniques with the BeautifulSoup library for HTML parsing, requests library for HTTP operations, and pandas for data manipulation. The tool is particularly useful for job seekers in the technology sector who want to efficiently gather and analyze remote job opportunities.

2. SOFTWARE DESCRIPTION

2.1 Python

Python is a high-level, versatile programming language known for its emphasis on code readability, which is achieved through significant indentation. It is dynamically typed and includes automatic garbage collection. Python supports various programming paradigms, such as structured (particularly procedural), object-oriented, and functional programming. Consistently ranked among the most popular programming languages, Python is especially prominent in the machine learning community.

According to the Stack Overflow 2025 Developer Survey, Python was the popular programming language among 31,771 respondents, with 57.9% reporting that they use it. And Python's adoption has accelerated significantly. It saw a 7%-point increase from 2024 to 2025

2.2 VS Code

Visual Studio Code, commonly known as VS Code, is a source-code editor developed by Microsoft that is available for Windows, Linux, macOS, and web browsers. It offers a range of features including debugging support, syntax highlighting, intelligent code completion, code snippets, refactoring capabilities, and integrated version control with Git. Users have the flexibility to customize themes, keyboard shortcuts, and preferences, as well as to install extensions to enhance functionality.

According to the Stack Overflow 2025 Developer Survey, Visual Studio Code was the most popular development environment tool among 26,143 respondents, with 75.9% reporting that they use it.

3. CODE SEGMENT DESCRIPTION

This project utilizes Requests, BeautifulSoup and Pandas packages to develop the web scraper and export data to CSV.

1. Requests - Handles HTTP requests to fetch web pages
2. BeautifulSoup - Parses HTML content and extracts data
3. Pandas - Manages data structures and CSV export functionality

Import Statements and Dependencies

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

Get the Search URL Function (`get_search_url`)

```
def get_search_url(keyword):
    if not keyword:
        raise ValueError("Keyword cannot be empty. Please provide a valid search
term.")
    else:
        keyword = keyword.strip().lower().replace(" ", "-")
        return f"https://remoteok.com/remote-{keyword}-jobs"
```

This function constructs the RemoteOK search URL based on user input. This function validates that the keyword is not empty, normalizes the keyword by converting to lowercase and replacing spaces with hyphens, and returns a properly formatted RemoteOK URL

Core Scraping Function (scrape_jobs)

```
def scrape_jobs(search_term):

    url = get_search_url(search_term)
    headers = {"User-Agent": "Mozilla/5.0"}

    response = requests.get(url, headers=headers)
    if response.status_code != 200:
        print("Failed to fetch data. Check the keyword or try again later.")
        return []
```

This function performs the actual web scraping and data extraction. Inside this function sets appropriate headers to mimic browser requests, handles HTTP response validation, Parses HTML content using BeautifulSoup, Extracts job information from table rows with class "job", and processes the extracted data.

Data Extraction Logic

```
for job in job_rows:
    company = job.get("data-company")
    position = job.find("td", class_="company_and_position").select("a > h2[itemprop='title']")[0].text.strip()
    date = job.find("td", class_="time").select("time")[0]["datetime"].strip()
    date = pd.to_datetime(date).strftime("%Y-%m-%d")
    tags = [tag.text.strip() for tag in job.find("td", class_="tags").select("a") if tag.text.strip()]
    link = job.get("data-url")
```

Main Function

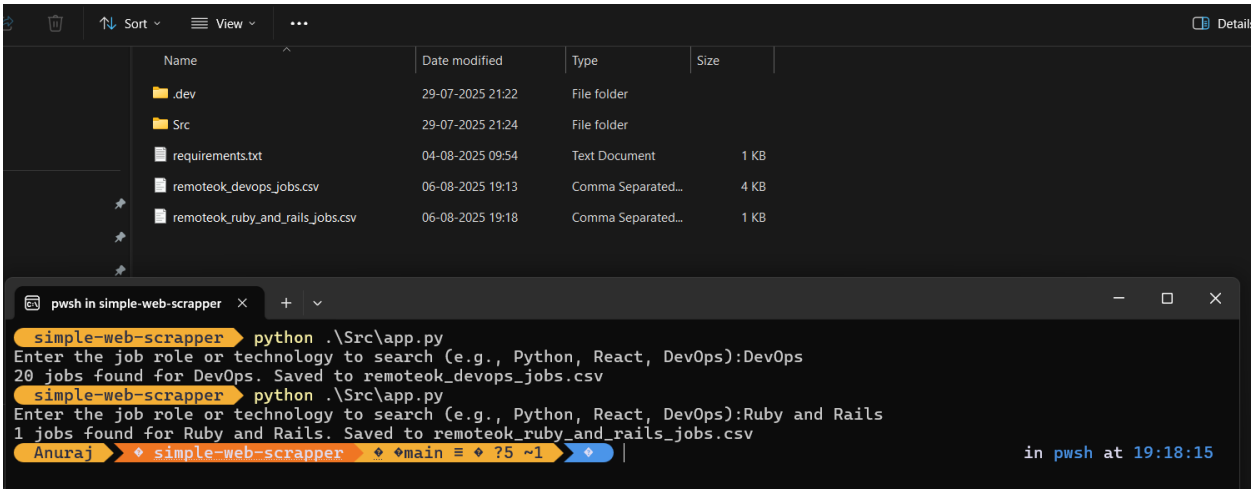
```
def main():
    search_input = input("Enter the job role or technology to search (e.g., Python, React, DevOps):")
    if not search_input.strip():
        print("No search term provided. Exiting.")
        return

    job_list = scrape_jobs(search_input)
    if job_list:
        df = pd.DataFrame(job_list)
        filename = f"remoteok_{search_input.lower().replace(' ', '_')}_jobs.csv"
        df.to_csv(filename, index=False)
        print(f"{len(df)} jobs found for {search_input}. Saved to {filename}")
    else:
        print("No jobs found or an error occurred.")
```

This function manages user interaction and coordinates the scraping process - it prompts user for search input, validates input and calls scraping function, creates pandas DataFrame from extracted data, exports data to CSV file with descriptive filename, provides user feedback on operation results

4. SAMPLE SCREENSHOTS

Fig 1: Application Running and Generating CSV



The application generates CSV files with the following structure:

Column	Description	Data Type
Company	Company name offering the position	String
Position	Job Title / Position name	String
Date Posted	Job Posting date (YYYY-MM-DD format)	Date
Tags	Technology stack and skills (max 5 tags)	List
Link	Direct link to the job posting	Url

5. CONCLUSION

The Simple Web Scraper successfully demonstrates the practical application of web scraping techniques for job market analysis. The tool effectively automates the time-consuming process of manual job searching by providing structured, exportable data that can be easily analyzed and processed. The application showcases several important programming concepts including HTTP request handling, HTML parsing, data manipulation, and file I/O operations. The modular design ensures maintainability and extensibility, making it an excellent foundation for more complex job scraping applications.

This project serves as both a practical tool for job seekers and an educational example of web scraping implementation in Python. The clean code structure and comprehensive error handling make it suitable for both beginners learning web scraping and professionals looking for a reliable job data extraction tool. The success of this project highlights the power of Python's ecosystem for web scraping tasks and demonstrates how relatively simple code can solve real-world problems efficiently.

6. BIBLIOGRAPHY

1. Beautiful Soup 4.13.4 Documentation - https://www.crummy.com/software/BeautifulSoup/bs4/doc/
2. Requests Library Documentation - https://requests.readthedocs.io/en/latest/
3. Pandas Documentation - https://pandas.pydata.org/docs/
4. RemoteOK - Remote Jobs in Programming, Design, Marketing and more - https://remoteok.io/