

**AMITY**  
**UNIVERSITY**  

---

**ONLINE**

**Project Report On**

**Automated Job Search Using Web Scraping**

**Submitted By:**

Name – Md. Azeem

MCA 3rd Semester

Enrollment No. - A9929722000768 (el)

## **ABSTRACT**

The purpose of this project is to automate the process of job searching by extracting relevant job postings from a specific website, TimesJobs. The project focuses on Python-related jobs and excludes those requiring a specific skill, Django. This automation aims to save job seekers time and effort, allowing them to focus on enhancing their skills and preparing for interviews. The project employs Python, a versatile programming language, and several of its libraries. BeautifulSoup is used for parsing HTML and XML documents, Requests for making HTTP requests, and Pandas for data manipulation and analysis. The script sends a GET request to the TimesJobs website, parses the HTML response to extract job postings, and saves the data in a structured format.

The outcome of the project is a successful extraction of job postings from the website, which are then stored in a CSV file and a text file. The CSV file contains all Python-related jobs, while the text file contains jobs that do not require Django. This demonstrates the potential of web scraping in automating mundane tasks and providing valuable insights from the data. The project also highlights the importance of ethical web scraping practices and the potential for customization based on the user's skill set. Future improvements could include adding more customization options and expanding the script to scrape job postings from multiple websites. In conclusion, this project showcases the power of Python and web scraping in automating the job search process, making it more efficient and less time-consuming. It also provides a foundation for future projects that aim to automate other aspects of job searching or other similar tasks.

## INDEX

<b>Sr. No.</b>	<b>CHAPTERS</b>	<b>PARTICULARS</b>
<b>1</b>	<b>CHAPTER 1</b>	<b>Introduction</b>
<b>2</b>	<b>CHAPTER 2</b>	<b>Objective of the Study</b>
<b>3</b>	<b>CHAPTER 3</b>	<b>Literature Review</b>
<b>4</b>	<b>CHAPTER 4</b>	<b>Research Methodology</b>
<b>5</b>	<b>CHAPTER 5</b>	<b>Data Analysis &amp; Interpretation</b>
<b>6</b>	<b>CHAPTER 6</b>	<b>Recommendations and Conclusion</b>
<b>7</b>	<b>CHAPTER 7</b>	<b>Bibliography &amp; References</b>

# CHAPTER 1

## INTRODUCTION

Web scraping is a technique for extracting vast volumes of data from websites. We can transform the unstructured data found on websites into a structured format by using web scraping. Web scraping is extremely significant in today's data-driven society since it enables us to utilize the enormous amount of data that is readily available online. This data can then be used for a variety of reasons, including data analysis, visualization, machine learning, and many more.

For a number of reasons, Python is a popular choice for web scraping:

1. **Ease of Use:** Python's straightforward syntax facilitates the writing and comprehension of code.
2. **Strong packages:** BeautifulSoup, Scrapy, and Selenium are just a few of the many web scraping packages available in Python. These libraries offer strong capabilities for a variety of tasks, including browsing and searching the parse tree, parsing HTML, and downloading webpages.
3. **Community Support:** Python has a large community of developers who contribute to its libraries and provide support through various forums.
4. **Versatility:** Python is useful for more than just web scraping. In addition, it can be applied to web development, machine learning, and data analysis. Python is therefore a flexible tool for handling the data you've scraped.

In conclusion, web scraping is a useful method in today's data-driven society, and Python is a great language for web scraping because of its robust libraries and ease of use.

## CHAPTER 2

### Objective of the Study

The specific objectives of this project are as follows:

1. **Automate the Job Search Process:** The primary objective is to automate the extraction of job postings from TimesJobs, a popular job search website. This saves job seekers time and effort, allowing them to focus on other important aspects such as enhancing their skills and preparing for interviews.
2. **Customize Job Search Based on Skills:** The code focuses on Python-related jobs and excludes those requiring Django. This allows users to customize their job search based on their skill set, increasing their chances of finding jobs that match their skills and preferences.
3. **Data Extraction and Analysis:** By extracting data such as the company name, location, skills required, and the link to the job posting, the script provides valuable data that can be analyzed to gain insights into the job market.

Extracting links from a website is of interest because these links represent job postings. Each link provides access to detailed information about a job posting, including the job description, qualifications required, company information, and application process. This information is crucial for job seekers as it helps them understand the job requirements and decide whether to apply.

The extracted data can provide several potential insights:

- **Job Market Trends:** By analyzing the skills required for the jobs, we can identify the most in-demand skills in the job market. This can help job seekers focus on acquiring the right skills.

- **Company Hiring Patterns:** The data can reveal which companies are hiring the most and what skills they are looking for. This can help job seekers target their applications effectively.
- **Geographical Insights:** By analyzing the location data, we can identify the regions with the most job opportunities. This can be useful for job seekers who are open to relocating.

In conclusion, the objectives of this project go beyond just automating the job search process. It also aims to provide valuable insights that can help job seekers navigate the job market more effectively.

## CHAPTER 3

### Literature Review

Web scraping has been used in numerous studies and projects for data extraction across various domains. Here are a few examples:

1. **Market Research:** Web scraping is commonly used in market research to gather data about products, prices, and competitors. For instance, a study might scrape e-commerce websites to analyze pricing strategies across different regions or categories. Tools like BeautifulSoup, Scrapy, and Selenium are often used in such projects.
2. **Social Media Analysis:** Studies have used web scraping to extract data from various social media platforms like Twitter, Facebook, and Instagram for sentiment analysis, trend spotting, and studying user behavior. Tweepy is a popular library for scraping Twitter data.
3. **Search Engine Optimization (SEO):** Web scraping is used to monitor website performance by extracting data about site traffic, ranking, and keyword performance. Libraries like BeautifulSoup and Requests, along with SEO tools like Moz and SEMrush, are often used.
4. **Academic Research:** In academic research, web scraping is used to gather data for a wide range of studies, from analyzing the spread of fake news to studying the impact of climate change on bird migration patterns.

Challenges in web scraping projects often include handling dynamic content on websites, dealing with rate limits and IP blocking, and navigating websites with complex structures. These challenges are typically addressed by:

- **Dynamic Content:** Websites with dynamic content load their data using JavaScript. Libraries like Selenium or tools like Puppeteer can interact with the JavaScript on the page to load and scrape this content.
- **Rate Limiting and IP Blocking:** To prevent IP blocking or rate limiting, web scrapers can rotate IP addresses using proxy servers, or add delays between requests to avoid making too many requests in a short period.
- **Complex Website Structures:** Navigating complex websites can be challenging. However, this can be overcome by studying the website's structure, often by inspecting the website's sitemap or using developer tools to understand the site's Document Object Model (DOM).

It's important to note that while web scraping can be a powerful tool, it's essential to respect the terms of service of the website being scraped and to scrape ethically.



## CHAPTER 4

### Research Methodology

The methodology of this project is as follows:

1. **Choice of Python and Libraries:** Python is chosen for this project due to its simplicity, versatility, and the powerful libraries it offers for web scraping and data manipulation.

The specific libraries used are:

- BeautifulSoup: A Python library used for parsing HTML and XML documents. It creates parse trees that are helpful to extract the data easily.
- requests: A Python library used for making HTTP requests. It allows you to concentrate on interacting with services and consuming data in your operation by abstracting the complexity of making requests behind a lovely, straightforward API.
- pandas: A Python library used for data manipulation and analysis. It is used to extract data and store it in a desired format.

2. **Code and Its Functionality:**

The code can be divided into two main functions: findJobs and findJobsFilter.

- findJobs function:
  - This function sends a GET request to the TimesJobs website using the requests.get() method.
  - The HTML content of the page is parsed using BeautifulSoup.
  - The function then finds all job postings on the page, which are HTML li elements with the class “clearfix job-bx wht-shd-bx”.
  - For each job posting, the function extracts the company name, location, skills required, and the link to the job posting.

- This data is stored in a dictionary, which is then appended to the jobs\_data list.
  - The function then creates a pandas DataFrame from jobs\_data and saves it to a CSV file.
  - The function returns the number of jobs found.
- findJobsFilter function:
  - This function is similar to findJobs, but with an additional filtering step.
  - After extracting the skills required for a job, the function checks if 'Django' is in the skills. If not, the job data is written to a text file.
  - The function returns the number of jobs found that don't require 'Django'.

In conclusion, this project demonstrates how Python and its libraries can be used to automate the job search process by scraping job postings from a website, filtering the jobs based on skills, and storing the data for further analysis. It showcases the power of web scraping in extracting valuable insights from web data and the potential of Python in automating such tasks.

## CHAPTER 5

### Data Analysis & Interpretation

The output of the program consists of two parts: a CSV file named `Jobs.csv`, `filterjobs.csv`, `Desiredjobs.csv` and a text file named `Pythonjobs_filter.txt`.

1. **Jobs.csv:** This CSV file contains all the Python-related jobs scraped from the TimesJobs website. Each row in the CSV file represents a job posting and includes the following information:

- Company: The name of the company that posted the job.
- Location: The location where the job is based.
- Skills: The skills required for the job.
- Link: The URL of the job posting on the TimesJobs website.

This list of links (URLs) in the CSV file represents the web pages where the job postings are detailed. By following these links, job seekers can find more information about the job postings, such as job descriptions, qualifications required, application procedures, etc.

2. **filterjobs.csv:** This CSV file contains all the Python-related jobs that has been filtered excluding Django as skill from the data scraped from the TimesJobs website.
3. **Pythonjobs\_filter.txt:** This text file contains Python-related jobs that do not require Django. The structure of the data is similar to the CSV file, but it's stored in a text format.

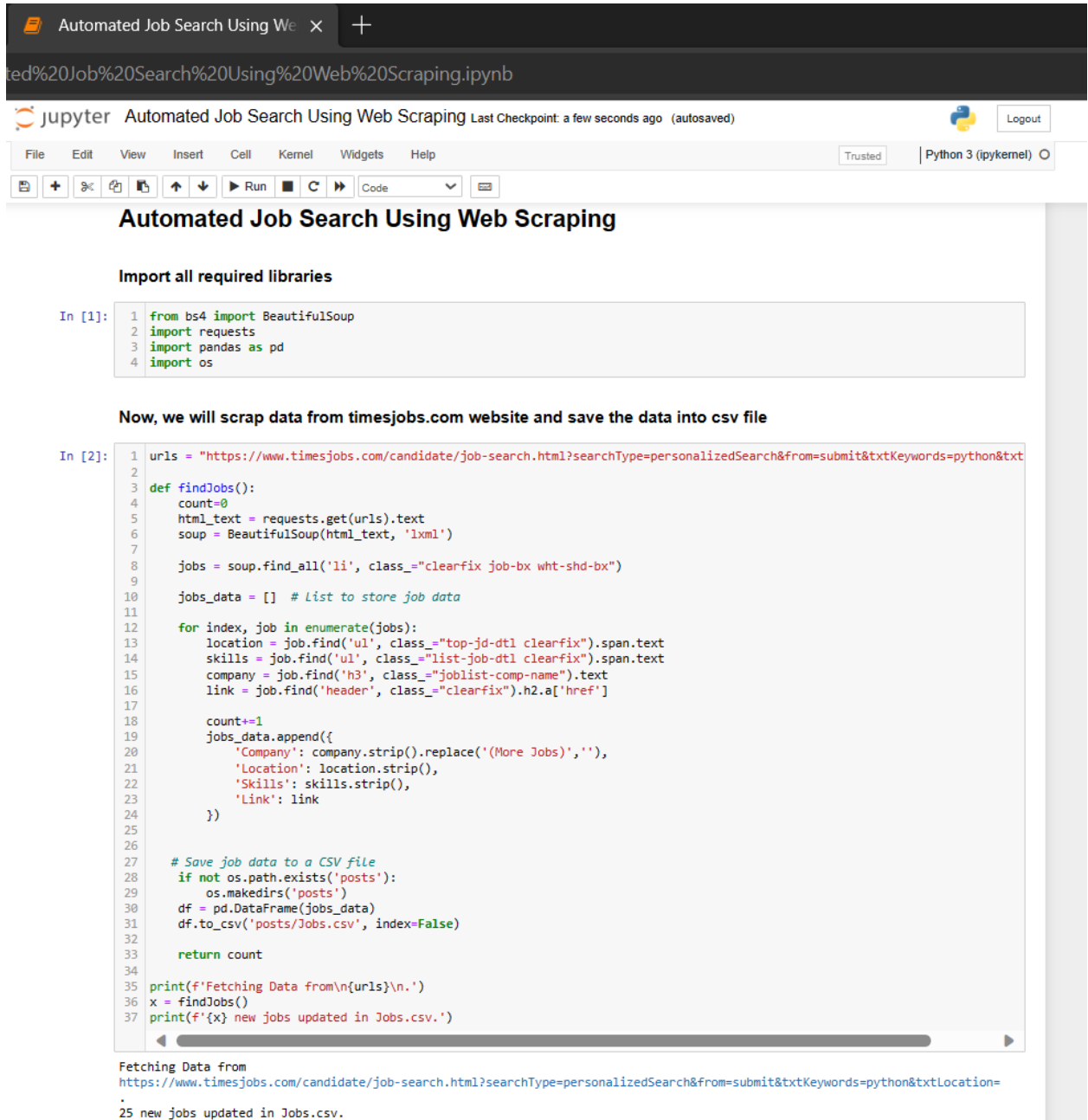
The data in these files can be interpreted or analyzed in several ways to gain insights into the job market. For example:

- **Skill Demand:** By analyzing the Skills column, you can identify which skills are most in demand for Python-related jobs. This can help job seekers focus on learning the right skills.
- **Job Availability:** The Location column can be analyzed to find out which locations have the most Python-related job opportunities. This can be useful for job seekers who are willing to relocate.
- **Company Hiring:** Analyzing the Company column can reveal which companies are hiring the most Python developers. Job seekers can target their applications to these companies.

In conclusion, the output of the program provides valuable data that can help job seekers navigate the job market more effectively. It also demonstrates the power of web scraping in extracting useful information from websites.

## Model Implementation:-

1. First of all, I imported all the required modules & libraries in the model.
2. Then, I scraped data from the Timesjobs.com and saved all the extracted data in the Jobs.csv file.



```
Automated Job Search Using Web Scraping Last Checkpoint: a few seconds ago (autosaved)
Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted

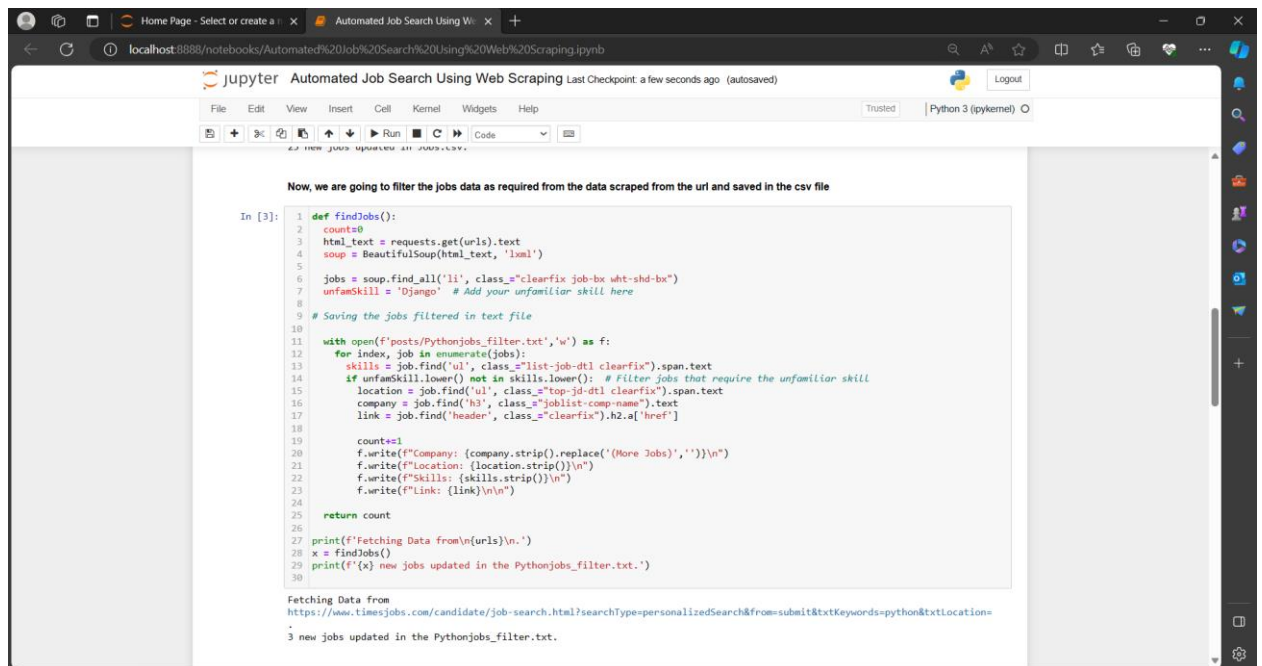
In [1]: 1 from bs4 import BeautifulSoup
        2 import requests
        3 import pandas as pd
        4 import os

Now, we will scrap data from timesjobs.com website and save the data into csv file

In [2]: 1 urls = "https://www.timesjobs.com/candidate/job-search.html?searchType=personalizedSearch&from=submit&txtKeywords=python&txtLocation="
        2
        3 def findJobs():
        4     count=0
        5     html_text = requests.get(urls).text
        6     soup = BeautifulSoup(html_text, 'lxml')
        7
        8     jobs = soup.find_all('li', class_="clearfix job-bx wht-shd-bx")
        9
        10    jobs_data = [] # List to store job data
        11
        12    for index, job in enumerate(jobs):
        13        location = job.find('ul', class_="top-jd-dtl clearfix").span.text
        14        skills = job.find('ul', class_="list-job-dtl clearfix").span.text
        15        company = job.find('h3', class_="joblist-comp-name").text
        16        link = job.find('header', class_="clearfix").h2.a['href']
        17
        18        count+=1
        19        jobs_data.append({
        20            'Company': company.strip().replace('More Jobs',''),
        21            'Location': location.strip(),
        22            'Skills': skills.strip(),
        23            'Link': link
        24        })
        25
        26    # Save job data to a CSV file
        27    if not os.path.exists('posts'):
        28        os.makedirs('posts')
        29    df = pd.DataFrame(jobs_data)
        30    df.to_csv('posts/Jobs.csv', index=False)
        31
        32    return count
        33
        34    print(f'Fetching Data from\n{urls}\n.')
        35    x = findJobs()
        36    print(f'{x} new jobs updated in Jobs.csv.')
        37

Fetching Data from
https://www.timesjobs.com/candidate/job-search.html?searchType=personalizedSearch&from=submit&txtKeywords=python&txtLocation=
.
25 new jobs updated in Jobs.csv.
```

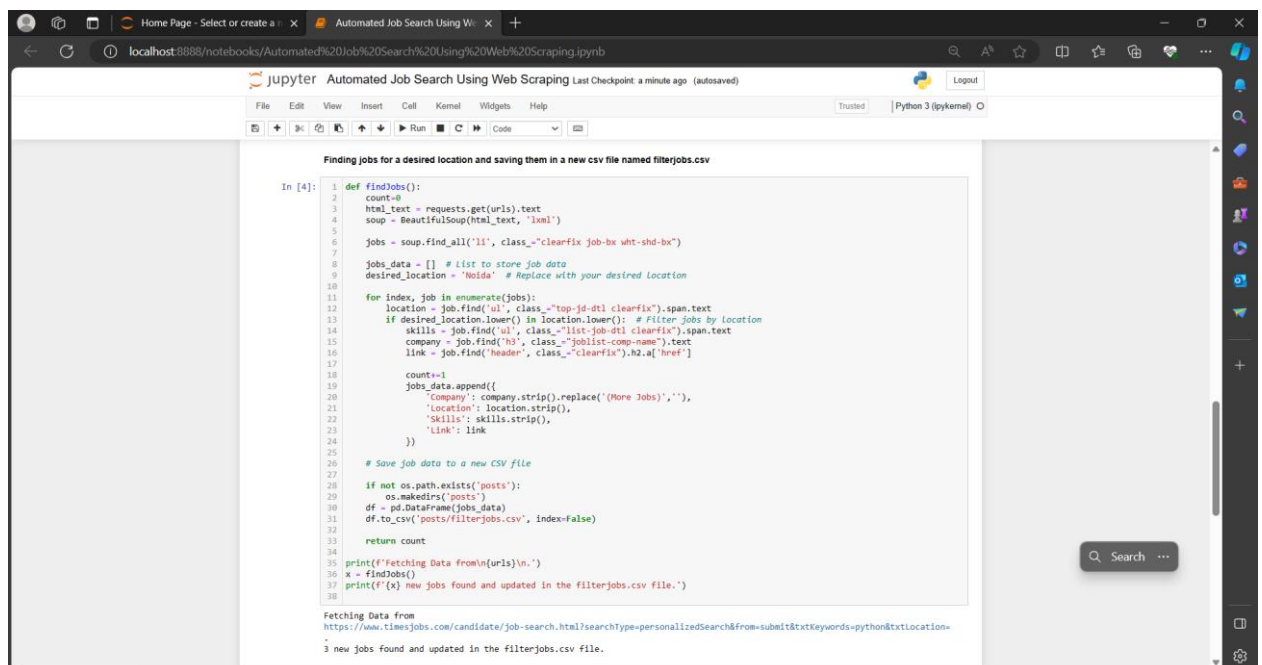
3. Then, I filtered the job data to exclude Django as a skill from the scraped data of the website, and stored the results in a txt file named Pythonjobs\_filter.txt



The screenshot shows a Jupyter Notebook interface with a single code cell. The code defines a function `findJobs()` that scrapes job data from a URL, filters out jobs with 'Django' in the skills, and saves the results to a text file named `Pythonjobs_filter.txt`. The output shows that 3 new jobs were updated in the file.

```
In [3]: def findJobs():
1 count=0
2 html_text = requests.get(urls).text
3 soup = BeautifulSoup(html_text, 'lxml')
4 jobs = soup.find_all('li', class_='clearfix job-bx wht-shd-bx')
5 unfaSkill = 'Django' # Add your unfamiliar skill here
6
7 # Saving the jobs filtered in text file
8
9 with open(f'posts/Pythonjobs_filter.txt', 'w') as f:
10     for index, job in enumerate(jobs):
11         skills = job.find('ul', class_='list-job-dtl clearfix').span.text
12         if unfaSkill.lower() not in skills.lower(): # Filter jobs that require the unfamiliar skill
13             location = job.find('ul', class_='top-jd-dtl clearfix').span.text
14             company = job.find('h3', class_='joblist-comp-name').text
15             link = job.find('header', class_='clearfix').h2.a['href']
16
17             count+=1
18             f.write(f'Company: {company.strip().replace(' (More Jobs)', '')}\n')
19             f.write(f'Location: {location.strip()}\n')
20             f.write(f'Skills: {skills.strip()}\n')
21             f.write(f'Link: {link}\n\n')
22
23     return count
24
25 print(f'Fetching Data from\n{urls}\n.')
26 x = findJobs()
27 print(f'{x} new jobs updated in the Pythonjobs_filter.txt.')
28
29 Fetching Data from
30 https://www.timesjobs.com/candidate/job-search.html?searchType=personalizedSearch&from=submit&txtKeywords=python&txtLocation=
31 3 new jobs updated in the Pythonjobs_filter.txt.
```

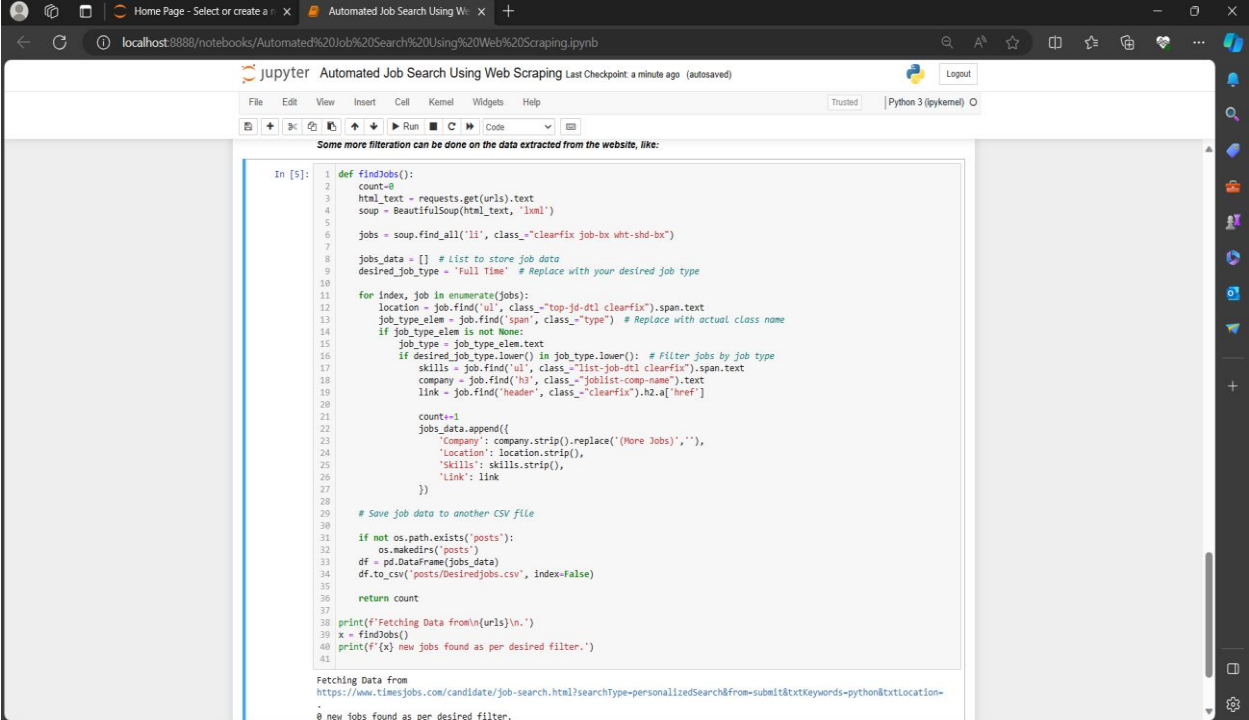
4. Now, I Filtered jobs for a desired location and saved them in a new csv file named filterjobs.csv.



The screenshot shows a Jupyter Notebook interface with a single code cell. The code defines a function `findJobs()` that scrapes job data from a URL, filters out jobs that do not match a desired location, and saves the results to a CSV file named `filterjobs.csv`. The output shows that 3 new jobs were found and updated in the file.

```
In [4]: def findJobs():
1 count=0
2 html_text = requests.get(urls).text
3 soup = BeautifulSoup(html_text, 'lxml')
4 jobs = soup.find_all('li', class_='clearfix job-bx wht-shd-bx')
5 jobs_data = [] # list to store job data
6 desired_location = 'Noida' # Replace with your desired location
7
8 for index, job in enumerate(jobs):
9     location = job.find('ul', class_='top-jd-dtl clearfix').span.text
10     if desired_location.lower() in location.lower(): # Filter jobs by location
11         skills = job.find('ul', class_='list-job-dtl clearfix').span.text
12         company = job.find('h3', class_='joblist-comp-name').text
13         link = job.find('header', class_='clearfix').h2.a['href']
14
15         count+=1
16         jobs_data.append({
17             'Company': company.strip().replace(' (More Jobs)', ''),
18             'Location': location.strip(),
19             'Skills': skills.strip(),
20             'Link': link
21         })
22
23 # Save job data to a new CSV file
24 if not os.path.exists('posts'):
25     os.makedirs('posts')
26 df = pd.DataFrame(jobs_data)
27 df.to_csv('posts/filterjobs.csv', index=False)
28
29 return count
30
31 print(f'Fetching Data from\n{urls}\n.')
32 x = findJobs()
33 print(f'{x} new jobs found and updated in the filterjobs.csv file.')
34
35 Fetching Data from
36 https://www.timesjobs.com/candidate/job-search.html?searchType=personalizedSearch&from=submit&txtKeywords=python&txtLocation=
37 3 new jobs found and updated in the filterjobs.csv file.
```

5. I did one more filtration based on job\_type option on the data and saved in another csv file named Desiredjobs.csv, many more filter can be done on the data extracted from the website.



The screenshot shows a Jupyter Notebook titled "Automated Job Search Using Web Scraping" running on a local host. The notebook contains a Python script that scrapes job data from a website and filters it based on a specified job type. The script uses BeautifulSoup for HTML parsing and pandas for data manipulation. It saves the filtered data to a CSV file named "Desiredjobs.csv".

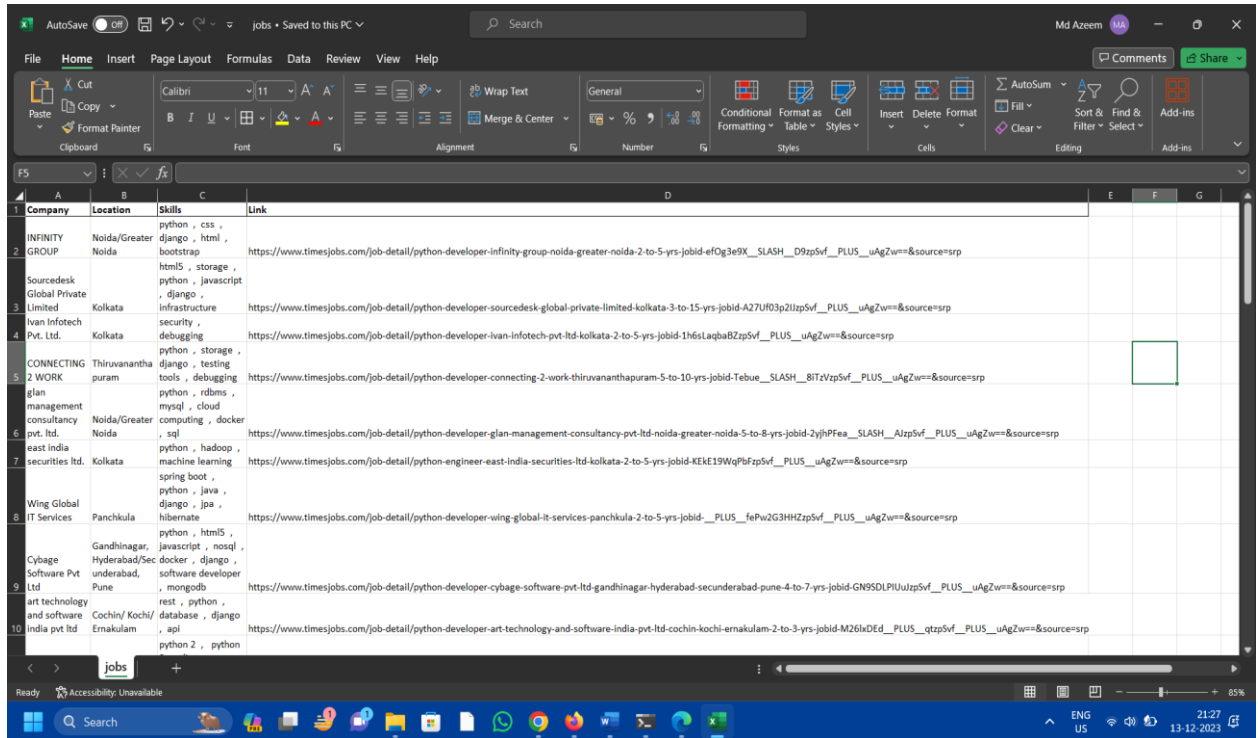
```
In [5]: 1 def findJobs():
2     count=0
3     html_text = requests.get(urls).text
4     soup = BeautifulSoup(html_text, 'lxml')
5
6     jobs = soup.find_all('li', class_="clearfix job-bx wht-shd-bx")
7
8     jobs_data = [] # list to store job data
9     desired_job_type = 'Full Time' # Replace with your desired job type
10
11     for index, job in enumerate(jobs):
12         location = job.find('ul', class_="top-jd-dtl clearfix").span.text
13         job_type_elem = job.find('span', class_="type") # Replace with actual class name
14         if job_type_elem is not None:
15             job_type = job_type_elem.text
16             if desired_job_type.lower() in job_type.lower(): # filter jobs by job type
17                 skills = job.find('ul', class_="list-job-dtl clearfix").span.text
18                 company = job.find('h3', class_="joblist-comp-name").text
19                 link = job.find('header', class_="clearfix").h2.a['href']
20
21                 count+=1
22                 jobs_data.append({
23                     'Company': company.strip().replace('More Jobs', ''),
24                     'Location': location.strip(),
25                     'Skills': skills.strip(),
26                     'Link': link
27                 })
28
29     # Save job data to another CSV file
30
31     if not os.path.exists('posts'):
32         os.makedirs('posts')
33     df = pd.DataFrame(jobs_data)
34     df.to_csv('posts/Desiredjobs.csv', index=False)
35
36     return count
37
38 print(f'Fetching Data from\n{urls}\n.')
39 x = findJobs()
40 print(f'{x} new jobs found as per desired filter.')
41
```

Fetching Data from  
https://www.timesjobs.com/candidate/job-search.html?searchType=personalizedSearch&from=submit&txtKeywords=python&txtLocation=  
0 new jobs found as per desired filter.

## OUTPUT:-

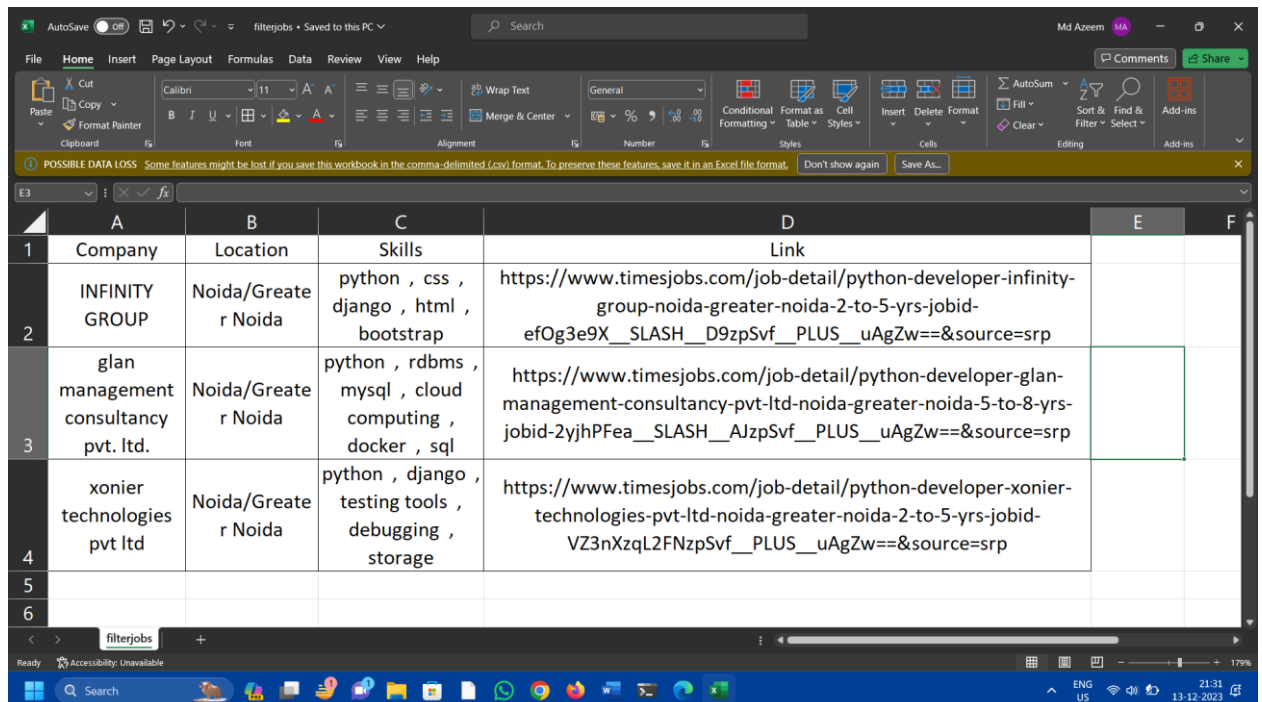
Data stored in Csv file :

### i. Jobs.csv



	A	B	C	D
	Company	Location	Skills	Link
1	INFINITY GROUP	Noida/Greater Noida	python , css , django , html , bootstrap	<a href="https://www.timesjobs.com/job-detail/python-developer-infinity-group-noida-greater-noida-2-to-5-yrs-jobid-efOg3e9X_SLASH_D9zpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-infinity-group-noida-greater-noida-2-to-5-yrs-jobid-efOg3e9X_SLASH_D9zpSvf_PLUS_uAgZw==&amp;source=srp</a>
2	Sourcedesk Global Private Limited	Kolkata	html5 , storage , python , javascript , django , infrastructure	<a href="https://www.timesjobs.com/job-detail/python-developer-sourcedesk-global-private-limited-kolkata-3-to-15-yrs-jobid-A27Uf03p2lzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-sourcedesk-global-private-limited-kolkata-3-to-15-yrs-jobid-A27Uf03p2lzpSvf_PLUS_uAgZw==&amp;source=srp</a>
3	Ivan Infotech Pvt. Ltd.	Kolkata	security , debugging	<a href="https://www.timesjobs.com/job-detail/python-developer-ivan-infotech-pvt-ltd-kolkata-2-to-5-yrs-jobid-1h6sLaqbaBZzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-ivan-infotech-pvt-ltd-kolkata-2-to-5-yrs-jobid-1h6sLaqbaBZzpSvf_PLUS_uAgZw==&amp;source=srp</a>
4	CONNECTING 2 WORK	Thiruvananthapuram	python , storage , django , testing tools , debugging	<a href="https://www.timesjobs.com/job-detail/python-developer-connecting-2-work-thiruvananthapuram-5-to-10-yrs-jobid-Tebue_SLASH_8ITzVzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-connecting-2-work-thiruvananthapuram-5-to-10-yrs-jobid-Tebue_SLASH_8ITzVzpSvf_PLUS_uAgZw==&amp;source=srp</a>
5	glan management consultancy pvt. ltd.	Noida/Greater Noida	python , rdbms , mysql , cloud computing , docker , sql	<a href="https://www.timesjobs.com/job-detail/python-developer-glan-management-consultancy-pvt-ltd-noida-greater-noida-5-to-8-yrs-jobid-2yjhPFea_SLASH_AJzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-glan-management-consultancy-pvt-ltd-noida-greater-noida-5-to-8-yrs-jobid-2yjhPFea_SLASH_AJzpSvf_PLUS_uAgZw==&amp;source=srp</a>
6	east india securities ltd.	Kolkata	python , hadoop , machine learning	<a href="https://www.timesjobs.com/job-detail/python-engineer-east-india-securities-ltd-kolkata-2-to-5-yrs-jobid-KEKE19WqBfzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-engineer-east-india-securities-ltd-kolkata-2-to-5-yrs-jobid-KEKE19WqBfzpSvf_PLUS_uAgZw==&amp;source=srp</a>
7	Wing Global IT Services	Panchkula	spring boot , python , java , django , jpa , hibernate	<a href="https://www.timesjobs.com/job-detail/python-developer-wing-global-it-services-panchkula-2-to-5-yrs-jobid-_PLUS_fePw2G3HHZzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-wing-global-it-services-panchkula-2-to-5-yrs-jobid-_PLUS_fePw2G3HHZzpSvf_PLUS_uAgZw==&amp;source=srp</a>
8	Cybage Software Pvt Ltd	Gandhinagar, Hyderabad/Secunderabad, Pune	python , html5 , javascript , nosql , docker , django , software developer	<a href="https://www.timesjobs.com/job-detail/python-developer-cybage-software-pvt-ltd-gandhinagar-hyderabad-secunderabad-pune-4-to-7-yrs-jobid-GN9SOLPIulzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-cybage-software-pvt-ltd-gandhinagar-hyderabad-secunderabad-pune-4-to-7-yrs-jobid-GN9SOLPIulzpSvf_PLUS_uAgZw==&amp;source=srp</a>
9	art technology and software india pvt ltd	Cochin/ Kochi/ Ernakulam	rest , python , database , django , api	<a href="https://www.timesjobs.com/job-detail/python-developer-art-technology-and-software-india-pvt-ltd-cochin-kochi-ernakulam-2-to-3-yrs-jobid-M26hDED_PLUS_qtZpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-art-technology-and-software-india-pvt-ltd-cochin-kochi-ernakulam-2-to-3-yrs-jobid-M26hDED_PLUS_qtZpSvf_PLUS_uAgZw==&amp;source=srp</a>
10			python 2 , python	

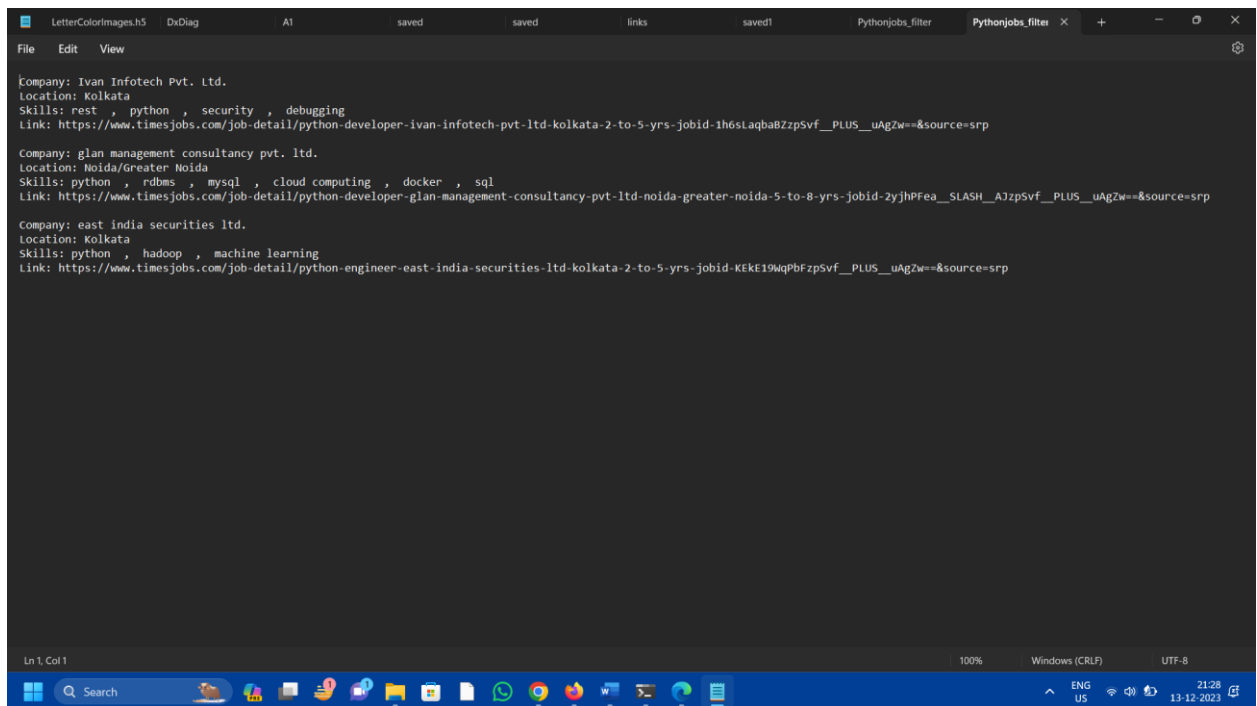
### ii. filterjobs.csv



	A	B	C	D
	Company	Location	Skills	Link
1	INFINITY GROUP	Noida/Greater Noida	python , css , django , html , bootstrap	<a href="https://www.timesjobs.com/job-detail/python-developer-infinity-group-noida-greater-noida-2-to-5-yrs-jobid-efOg3e9X_SLASH_D9zpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-infinity-group-noida-greater-noida-2-to-5-yrs-jobid-efOg3e9X_SLASH_D9zpSvf_PLUS_uAgZw==&amp;source=srp</a>
2	glan management consultancy pvt. ltd.	Noida/Greater Noida	python , rdbms , mysql , cloud computing , docker , sql	<a href="https://www.timesjobs.com/job-detail/python-developer-glan-management-consultancy-pvt-ltd-noida-greater-noida-5-to-8-yrs-jobid-2yjhPFea_SLASH_AJzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-glan-management-consultancy-pvt-ltd-noida-greater-noida-5-to-8-yrs-jobid-2yjhPFea_SLASH_AJzpSvf_PLUS_uAgZw==&amp;source=srp</a>
3	xonier technologies pvt ltd	Noida/Greater Noida	python , django , testing tools , debugging , storage	<a href="https://www.timesjobs.com/job-detail/python-developer-xonier-technologies-pvt-ltd-noida-greater-noida-2-to-5-yrs-jobid-VZ3nXzqL2FNzpSvf_PLUS_uAgZw==&amp;source=srp">https://www.timesjobs.com/job-detail/python-developer-xonier-technologies-pvt-ltd-noida-greater-noida-2-to-5-yrs-jobid-VZ3nXzqL2FNzpSvf_PLUS_uAgZw==&amp;source=srp</a>
4				
5				
6				



## Data Stored in Pythonjobs\_filter.txt file :



```
File Edit View

Company: Ivan Infotech Pvt. Ltd.
Location: Kolkata
Skills: rest , python , security , debugging
Link: https://www.timesjobs.com/job-detail/python-developer-ivan-infotech-pvt-ltd-kolkata-2-to-5-yrs-jobid-1h6sIaqbaBZzpsvf__PLUS__uAgZw==&source=srp

Company: glan management consultancy pvt. ltd.
Location: Noida/Greater Noida
Skills: python , rdms , mysql , cloud computing , docker , sql
Link: https://www.timesjobs.com/job-detail/python-developer-glan-management-consultancy-pvt-ltd-noida-greater-noida-5-to-8-yrs-jobid-2yjHPFea__SLASH__A3zpsvf__PLUS__uAgZw==&source=srp

Company: east india securities ltd.
Location: Kolkata
Skills: python , hadoop , machine learning
Link: https://www.timesjobs.com/job-detail/python-engineer-east-india-securities-ltd-kolkata-2-to-5-yrs-jobid-KEkE19WqPbFzpsvf__PLUS__uAgZw==&source=srp

Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

## CHAPTER 6

### Recommendations & Conclusion

Based on the project's findings, the following recommendations can be made:

1. **Skill Development:** Job seekers, especially those interested in Python-related jobs, should focus on developing the skills most in demand in the job market. The data extracted in this project can provide valuable insights into the most sought-after skills.
2. **Customization:** The script can be further customized to filter jobs based on other criteria such as location, company, or other skills. This would make the job search process even more tailored to the user's needs.
3. **Expansion:** The script currently focuses on one website, TimesJobs. It could be expanded to include other job search websites, providing a more comprehensive view of the job market.

The implications of this project are significant. It demonstrates the power of web scraping in automating a task that is traditionally done manually, saving time and effort. The project also shows how web scraping can extract valuable insights from web data, which can inform decision-making in various contexts.

Potential applications of this project extend beyond job searching. Similar techniques could be used to automate other types of online searches, such as house hunting, price comparison, sentiment analysis, and more. The possibilities are vast and limited only by the ethical considerations of web scraping.

In conclusion, this project showcases the power of Python and web scraping in automating the job search process. It demonstrates how a simple Python script can transform a tedious task into an easy and efficient one. The project also highlights the potential of web scraping in

extracting valuable insights from web data, providing a foundation for future projects in this field. The significance of this project lies not only in its immediate application to job searching but also in its broader implications for web scraping and automation.

## CHAPTER 7

### Bibliography & References

1. Python Software Foundation. (2023). Python Documentation. Retrieved from <https://docs.python.org/3/>
2. Richardson, L., Beautiful Soup. (2023). Beautiful Soup Documentation. Retrieved from <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
3. Reitz, K., & Schlusser, T. (2023). Requests: HTTP for Humans. Retrieved from <https://docs.python-requests.org/en/latest/>
4. The pandas development team. (2023). pandas: powerful Python data analysis toolkit. Retrieved from <https://pandas.pydata.org/docs/>