Indeed Job Scraper

1. Introduction

Searching for jobs online has become an essential step for students, professionals, and recruiters. However, manually browsing job boards like **Indeed** is repetitive and time-consuming. Job seekers may need to analyze trends such as which companies are hiring more, which cities have more opportunities, or which skills are in demand.

The **Indeed Job Scraper with Simple Analytics** is a Python-based mini project designed to automate this process. It extracts job details from Indeed and generates **data-driven insights** using charts and word clouds.

2. Problem Statement

- Job seekers manually check job boards every day, wasting time and effort.
- Recruiters and students lack quick tools to identify **hiring companies, locations, and skill trends**.
- A simple scraper can solve this problem but must also provide **unique features** to be useful.

3. Objectives

The objectives of this project are:

1. Automate job search data collection from **Indeed**.
2. Extract **job title, company, location, date posted, description, and job link**.
3. Save the results in **CSV/Excel format** for further use.
4. Provide **extra features** such as:

- Visualization of **top hiring companies**.
- **Jobs by location** analysis.
- A **Word Cloud** to highlight trending skills from job descriptions.
- **Simple filtering** of jobs by company or location.

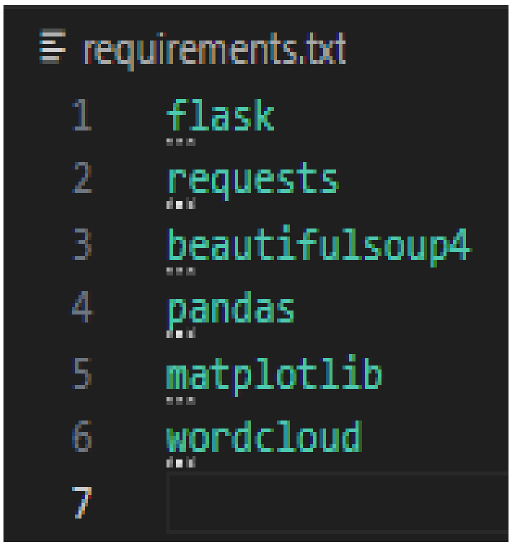
4. Tools and Technologies

4.1 Programming Language

- **Python 3.9+**

4.2 Libraries Used

- **requests** → Fetch HTML pages from Indeed.
- **beautifulsoup4** → Parse and extract job details from HTML.
- **pandas** → Store, clean, and export results into CSV/Excel.
- **matplotlib** → Generate bar charts for analytics.
- **wordcloud** → Create word clouds of skills/keywords from job descriptions.



requirements.txt

```
flask
requests
beautifulsoup4
pandas
matplotlib
wordcloud
```

5. System Design

5.1 Workflow

```
User Input (Keyword + Location)
        ↓
Requests → BeautifulSoup
        ↓
Extract job details (Title, Company, Location, Date, Description, URL)
        ↓
Save in Pandas DataFrame
        ↓
Export as CSV/Excel
        ↓
Analytics
   - Top companies (bar chart)
   - Jobs by location (bar chart)
   - Skill cloud (word cloud)
        ↓
Optional: Filter jobs by company/location
```

5.2 Data Model (columns saved in CSV)

- Job Title
- Company
- Location
- Date Posted
- Job Description
- Job URL

6. Implementation

6.1 Data Collection

- Construct the Indeed search URL using keyword and location.
- Use requests to fetch job listing pages.
- Parse HTML with BeautifulSoup to extract details.

- Loop through 2–3 pages using pagination.

6.2 Data Storage

- Store scraped job data in a pandas DataFrame.
- Export results into:
  - jobs.csv (CSV format)
  - jobs.xlsx (Excel format, optional)

6.3 Data Analytics (Unique Features)

1. Top Hiring Companies
   - Count job postings by company.
   - Plot a bar chart of top 5 companies.
2. Jobs by Location
   - Count job postings by city/region.
   - Plot a bar chart of top 10 locations.
3. Skill Word Cloud
   - Combine all job descriptions.
   - Generate a word cloud showing frequent terms (e.g., *Python, SQL, Excel*).
4. Simple Filtering
   - Allow users to filter the CSV results by company or location.

---

7. Sample Outputs

7.1 CSV File (jobs.csv)

Job Title	Company	Location	Date Posted	Description	Job URL
Data Analyst	TCS	Chennai	2 days ago	Analyze data...	…
Python Developer	Infosys	Bangalore	5 days ago	Develop Python apps...	…

7.2 Visualization (Matplotlib Charts)

- Top 5 Hiring Companies – Bar chart showing which companies have the most postings.
- Jobs by Location – Bar chart showing job counts in different cities.

7.3 Word Cloud (Skills)

- Word cloud image generated from job descriptions.
- Highlights common skills like Python, SQL, Excel, Machine Learning.

---

8. Results

- Successfully scraped job data from Indeed for a chosen keyword and location.
- Stored structured job data in CSV format.
- Generated insights:
    - Top companies hiring.
    - Job distribution by location.
    - Trending skills in demand.
- Added filtering for customized job search.

---

9. Applications

- Students: Identify trending skills for placements.
- Job Seekers: Save time by analyzing multiple job postings at once.
- Recruiters: Observe competitor hiring trends.
- Researchers: Study market demand across industries.

---

10. Limitations

- Some jobs may not have salary or full descriptions available.
- Website HTML structure may change (scraper must be updated).
- Heavy scraping may trigger rate limits

---

11. Future Enhancements

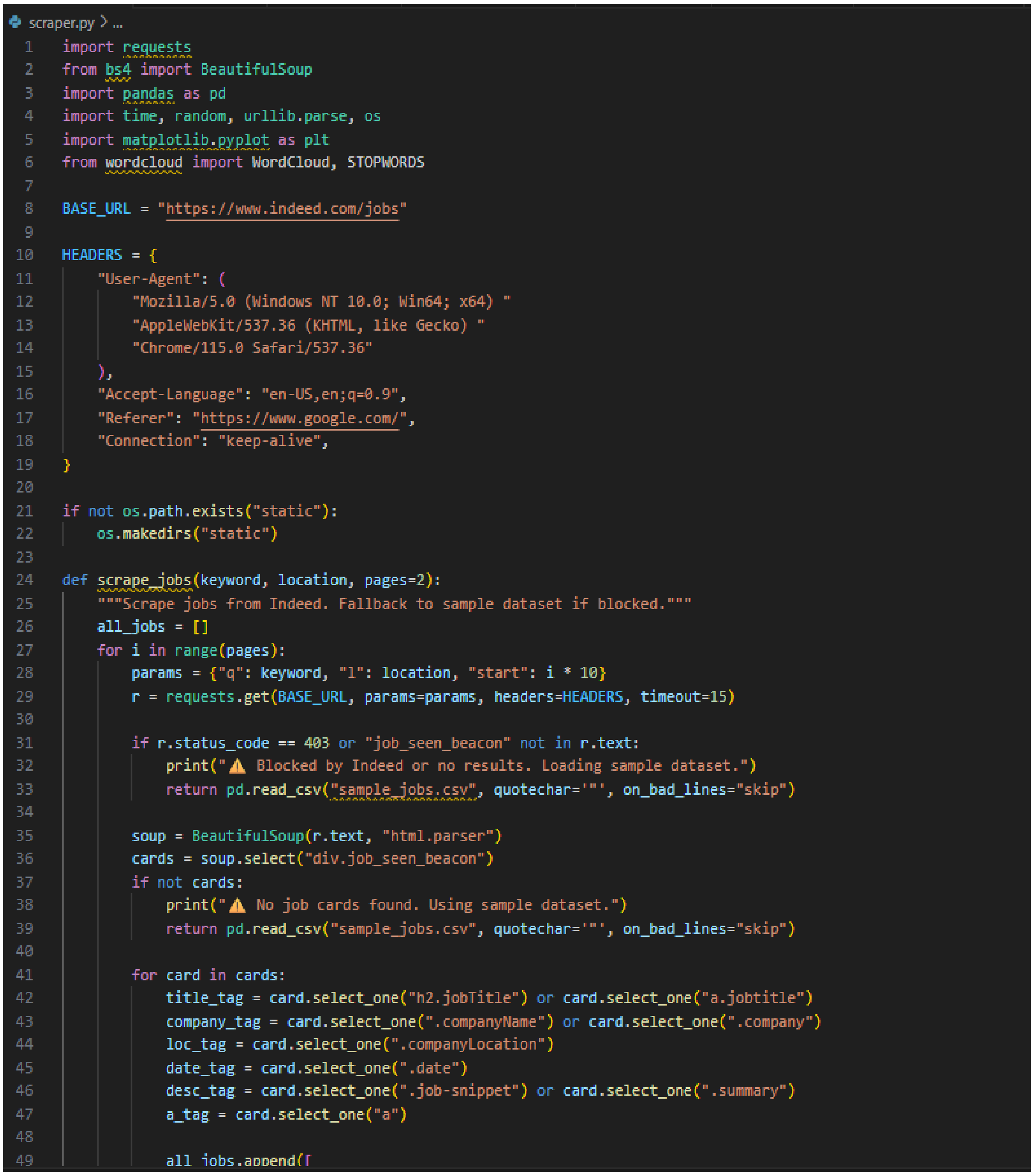
- Add salary extraction for salary trend analysis.
- Use Selenium for dynamic scraping.
- Automate scraping daily using schedule/cron jobs.
- Build a Streamlit web app for interactive job search.
- Compare multiple roles (e.g., Data Analyst vs Software Engineer).

12. Conclusion

The Indeed Job Scraper with Simple Analytics is a practical mini project that goes beyond plain web scraping. By combining data extraction with visualization and word clouds, it transforms job postings into actionable insights. This project is simple enough to implement yet unique enough to stand out.

Appendix

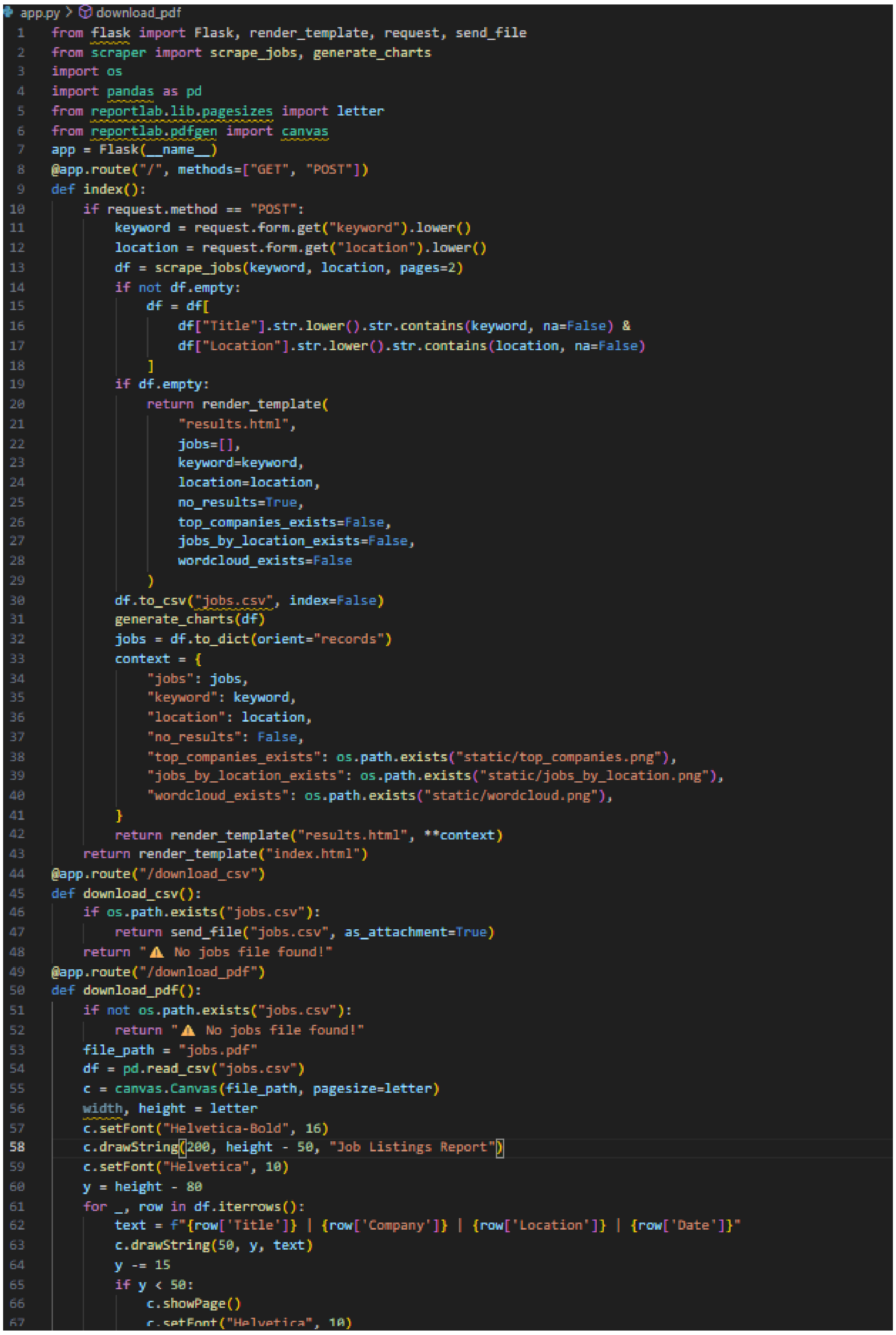
Scarper.py



scraper.py > ...

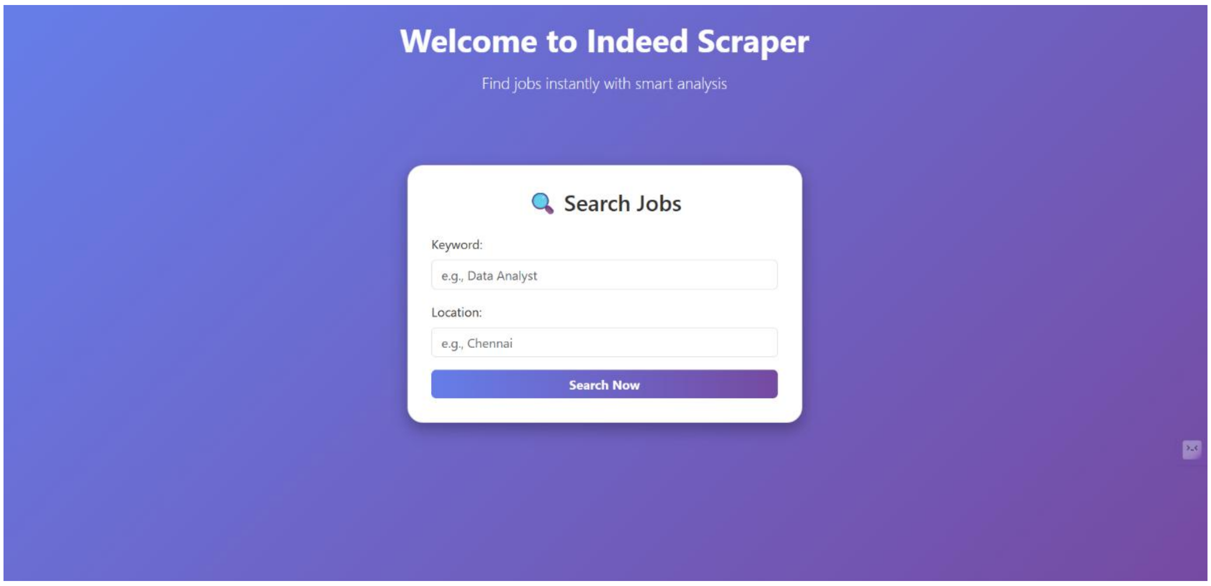
```python
import requests
from bs4 import BeautifulSoup
import pandas as pd
import time, random, urllib.parse, os
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS

BASE_URL = "https://www.indeed.com/jobs"

HEADERS = {
    "User-Agent": (
        "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
        "AppleWebKit/537.36 (KHTML, like Gecko) "
        "Chrome/115.0 Safari/537.36"
    ),
    "Accept-Language": "en-US,en;q=0.9",
    "Referer": "https://www.google.com/",
    "Connection": "keep-alive",
}

if not os.path.exists("static"):
    os.makedirs("static")

def scrape_jobs(keyword, location, pages=2):
    """Scrape jobs from Indeed. Fallback to sample dataset if blocked."""
    all_jobs = []
    for i in range(pages):
        params = {"q": keyword, "l": location, "start": i * 10}
        r = requests.get(BASE_URL, params=params, headers=HEADERS, timeout=15)

        if r.status_code == 403 or "job_seen_beacon" not in r.text:
            print("⚠ Blocked by Indeed or no results. Loading sample dataset.")
            return pd.read_csv("sample_jobs.csv", quotechar='"', on_bad_lines="skip")

        soup = BeautifulSoup(r.text, "html.parser")
        cards = soup.select("div.job_seen_beacon")
        if not cards:
            print("⚠ No job cards found. Using sample dataset.")
            return pd.read_csv("sample_jobs.csv", quotechar='"', on_bad_lines="skip")

        for card in cards:
            title_tag = card.select_one("h2.jobTitle") or card.select_one("a.jobtitle")
            company_tag = card.select_one(".companyName") or card.select_one(".company")
            loc_tag = card.select_one(".companyLocation")
            date_tag = card.select_one(".date")
            desc_tag = card.select_one(".job-snippet") or card.select_one(".summary")
            a_tag = card.select_one("a")

            all_jobs.append({
```

App.py



```python
from flask import Flask, render_template, request, send_file
from scraper import scrape_jobs, generate_charts
import os
import pandas as pd
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
app = Flask(__name__)
@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        keyword = request.form.get("keyword").lower()
        location = request.form.get("location").lower()
        df = scrape_jobs(keyword, location, pages=2)
        if not df.empty:
            df = df[
                df["Title"].str.lower().str.contains(keyword, na=False) &
                df["Location"].str.lower().str.contains(location, na=False)
            ]
        if df.empty:
            return render_template(
                "results.html",
                jobs=[],
                keyword=keyword,
                location=location,
                no_results=True,
                top_companies_exists=False,
                jobs_by_location_exists=False,
                wordcloud_exists=False
            )
        df.to_csv("jobs.csv", index=False)
        generate_charts(df)
        jobs = df.to_dict(orient="records")
        context = {
            "jobs": jobs,
            "keyword": keyword,
            "location": location,
            "no_results": False,
            "top_companies_exists": os.path.exists("static/top_companies.png"),
            "jobs_by_location_exists": os.path.exists("static/jobs_by_location.png"),
            "wordcloud_exists": os.path.exists("static/wordcloud.png"),
        }
        return render_template("results.html", **context)
    return render_template("index.html")
@app.route("/download_csv")
def download_csv():
    if os.path.exists("jobs.csv"):
        return send_file("jobs.csv", as_attachment=True)
    return "⚠ No jobs file found!"
@app.route("/download_pdf")
def download_pdf():
    if not os.path.exists("jobs.csv"):
        return "⚠ No jobs file found!"
    file_path = "jobs.pdf"
    df = pd.read_csv("jobs.csv")
    c = canvas.Canvas(file_path, pagesize=letter)
    width, height = letter
    c.setFont("Helvetica-Bold", 16)
    c.drawString(200, height - 50, "Job Listings Report")
    c.setFont("Helvetica", 10)
    y = height - 80
    for _, row in df.iterrows():
        text = f"{row['Title']} | {row['Company']} | {row['Location']} | {row['Date']}"
        c.drawString(50, y, text)
        y -= 15
        if y < 50:
            c.showPage()
            c.setFont("Helvetica", 10)
```

Output:



Welcome to Indeed Scraper

Find jobs instantly with smart analysis

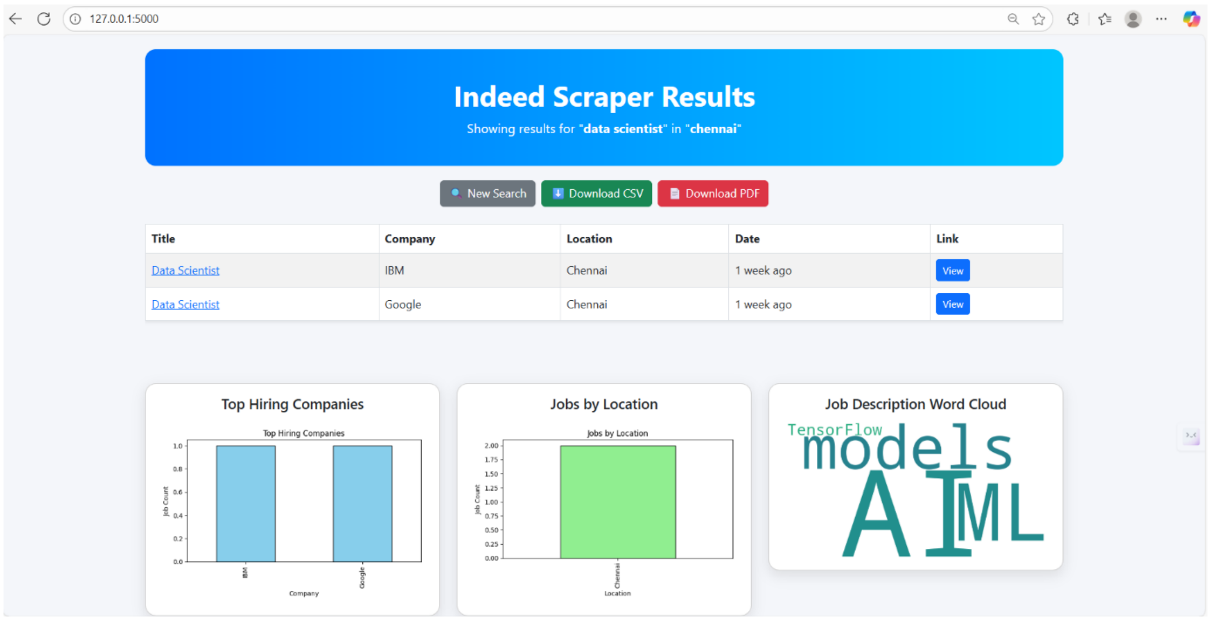
Search Jobs

Keyword:

e.g., Data Analyst

Location:

e.g., Chennai



Indeed Scraper Results

Showing results for "**data scientist**" in "**chennai**"

Title	Company	Location	Date	Link
Data Scientist	IBM	Chennai	1 week ago	View
Data Scientist	Google	Chennai	1 week ago	View

Cybernaut mini project-1

TEAM-6
MAHALAKSHMI S