

TECH JOB PORTAL BY SCRAPING INDEED AND GLASSDOOR

-Krishna Gowsalya M

INTRODUCTION

In the modern employment landscape, countless job listings are posted daily across various websites. Navigating through these listings to find relevant tech jobs can be time-consuming. Traditional job portals often include roles from all industries, making it difficult for developers and IT professionals to locate opportunities that match their skills.

The Tech Job Portal addresses this issue by focusing exclusively on technology-based jobs scraped directly from Indeed. Using automated scripts, it extracts job titles, company names, locations, and application links. The data is then displayed in a clean, user-friendly web interface built with Flask.

The portal provides an efficient and organized way to explore tech-related opportunities without the distractions of unrelated job listings.

OBJECTIVES

- To create a web portal dedicated to technology-related job listings.
- To automatically extract job postings from Indeed using web scraping.
- To develop a simple user system with login and registration features.
- To use a database (SQLite) for storing and retrieving job data.
- To display job details (title, company, location, and apply link) dynamically.
- To provide an easy-to-navigate interface for users to explore jobs.
- To simulate real-time job fetching for demonstration purposes.

LIBRARIES USED

The Tech Job Portal project utilizes several important Python libraries to perform web scraping, data storage, and web application development efficiently. Each library plays a specific role in the functioning of the system.

1.Flask

Flask is a lightweight and flexible web framework used to build the backend of the project. It handles routing, template rendering, and user session management. Flask makes it easy to connect the web interface with the Python logic and database.

2.Requests

The Requests library is used to send HTTP requests to the Indeed website. It allows the scraper to fetch the HTML content of job listing pages easily and reliably.

3.Beautifulsoup

BeautifulSoup, part of the bs4 module, is used for parsing and extracting job-related data from the HTML content fetched from Indeed. It helps identify specific HTML tags containing job titles, companies, locations, and links.

SYSTEM REQUIREMENTS

Hardware Requirements

Processor: Intel Core i3 or higher RAM:

Minimum 4 GB Hard Disk: Minimum

500 MB free space Display: Standard 15-
inch or higher

Software Requirements

- ❑ Programming Language: Python

Software Requirements

Programming Language: Python

Framework: Flask

Libraries Used: BeautifulSoup, Requests, SQLite3

Tools: Visual Studio Code

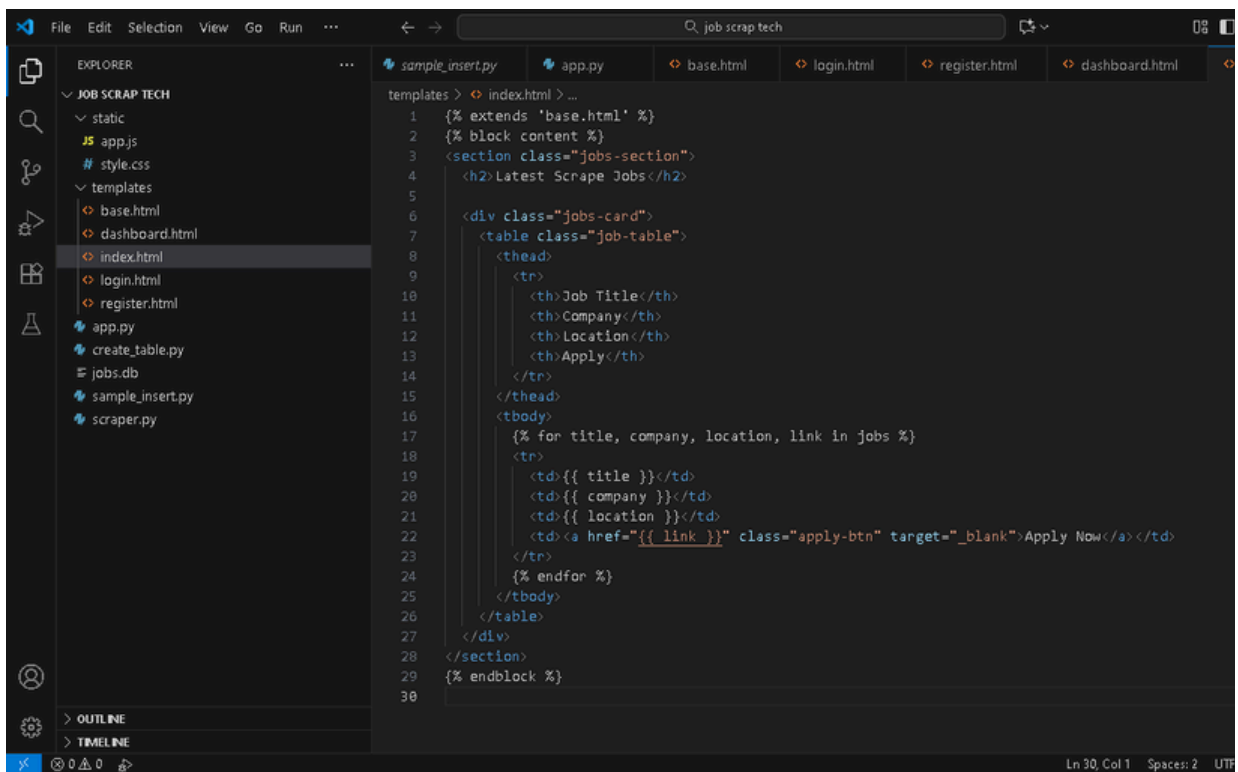
SYSTEM DESIGN

The project is structured using a three-tier architecture:

Frontend:

Handles user interaction using HTML, CSS, and JavaScript. Provides forms for login, registration, and job search.

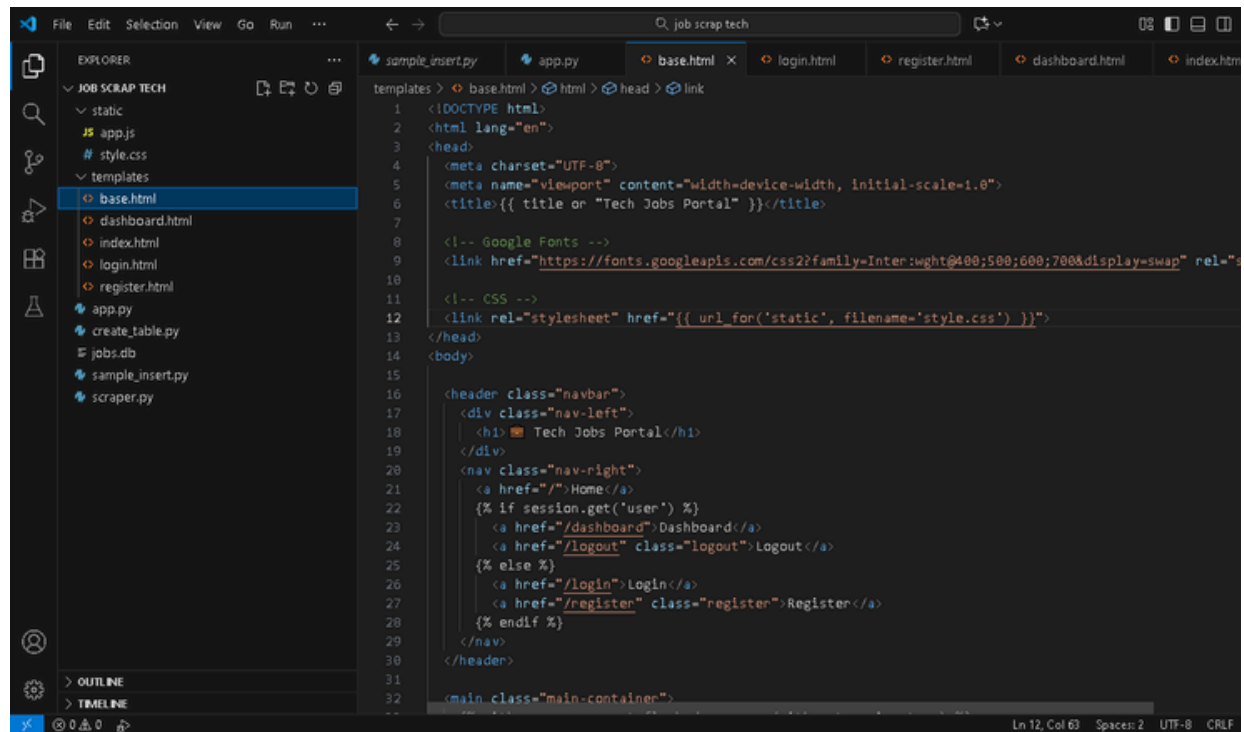
Index.html



The screenshot shows the Visual Studio Code interface with the 'JOB SCRAP TECH' project. The Explorer sidebar on the left lists the project files: static, app.js, style.css, templates, base.html, dashboard.html, index.html, login.html, register.html, app.py, create_table.py, jobs.db, sample_insert.py, and scraper.py. The main editor area displays the content of 'index.html', which is a Jinja2 template. It extends 'base.html' and contains a section for 'Latest Scrape Jobs'. This section includes a table with columns for Job Title, Company, Location, and Apply. The table body uses a loop to iterate over jobs, displaying their details and an 'Apply Now' button. The status bar at the bottom indicates the cursor is at line 30, column 1, with 2 spaces and UTF-8 encoding.

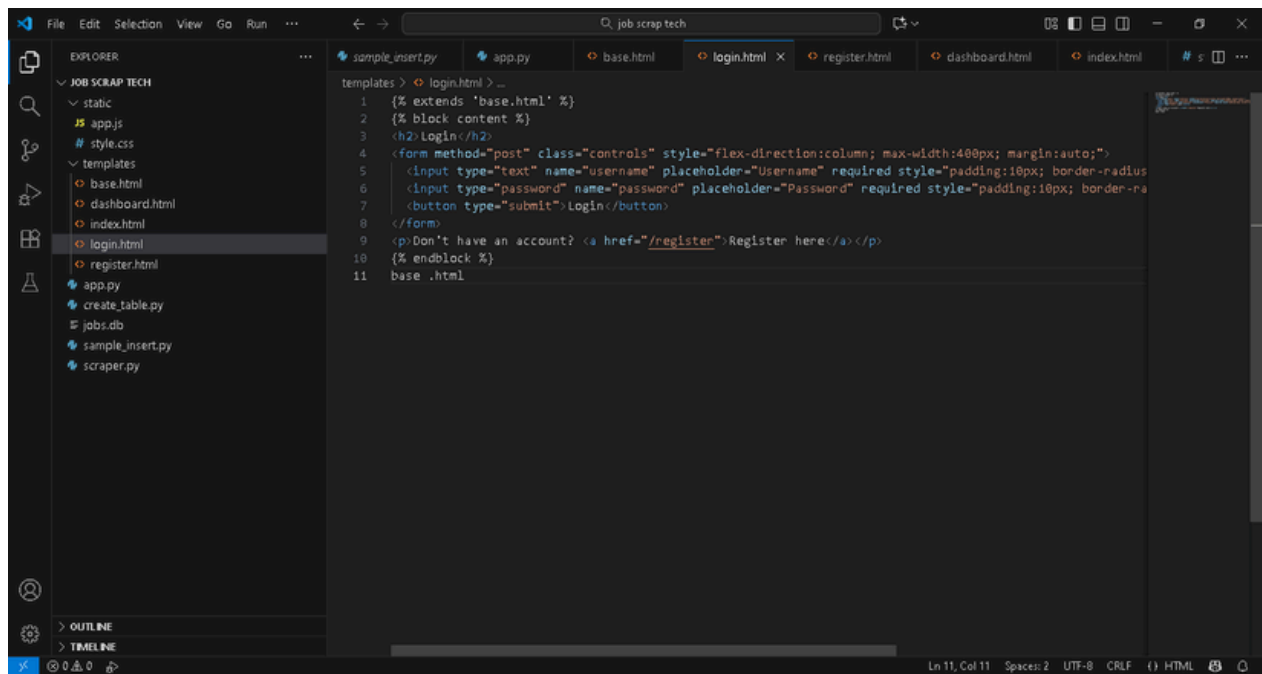
```
1 {% extends 'base.html' %}
2 {% block content %}
3 <section class="jobs-section">
4   <h2>Latest Scrape Jobs</h2>
5
6   <div class="jobs-card">
7     <table class="job-table">
8       <thead>
9         <tr>
10          <th>Job Title</th>
11          <th>Company</th>
12          <th>Location</th>
13          <th>Apply</th>
14        </tr>
15      </thead>
16      <tbody>
17        {% for title, company, location, link in jobs %}
18        <tr>
19          <td>{{ title }}</td>
20          <td>{{ company }}</td>
21          <td>{{ location }}</td>
22          <td><a href="{{ link }}" class="apply-btn" target="_blank">Apply Now</a></td>
23        </tr>
24        {% endfor %}
25      </tbody>
26    </table>
27  </div>
28 </section>
29 {% endblock %}
30
```

base.html



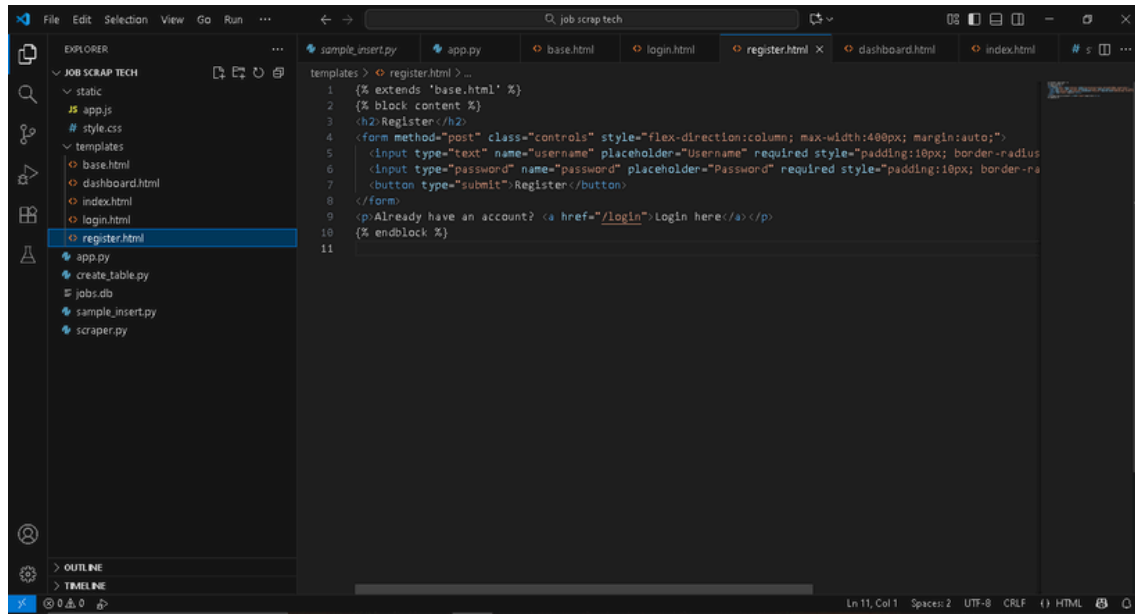
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>{{ title or "Tech Jobs Portal" }}</title>
7
8   <!-- Google Fonts -->
9   <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&display=swap" rel="stylesheet">
10
11   <!-- CSS -->
12   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
13 </head>
14 <body>
15
16   <header class="navbar">
17     <div class="nav-left">
18       <h1> Tech Jobs Portal</h1>
19     </div>
20     <nav class="nav-right">
21       <a href="/">Home</a>
22       {% if session.get('user') %}
23       <a href="/dashboard">Dashboard</a>
24       <a href="/logout" class="logout">Logout</a>
25       {% else %}
26       <a href="/login">Login</a>
27       <a href="/register" class="register">Register</a>
28       {% endif %}
29     </nav>
30   </header>
31
32   <main class="main-container">
```

login.html



```
1 {% extends 'base.html' %}
2 {% block content %}
3   <h2>Login</h2>
4   <form method="post" class="controls" style="flex-direction:column; max-width:400px; margin:auto;">
5     <input type="text" name="username" placeholder="Username" required style="padding:10px; border-radius
6     <input type="password" name="password" placeholder="Password" required style="padding:10px; border-ra
7     <button type="submit">Login</button>
8   </form>
9   <p>Don't have an account? <a href="/register">Register here</a></p>
10 {% endblock %}
11 base.html
```

Register.html

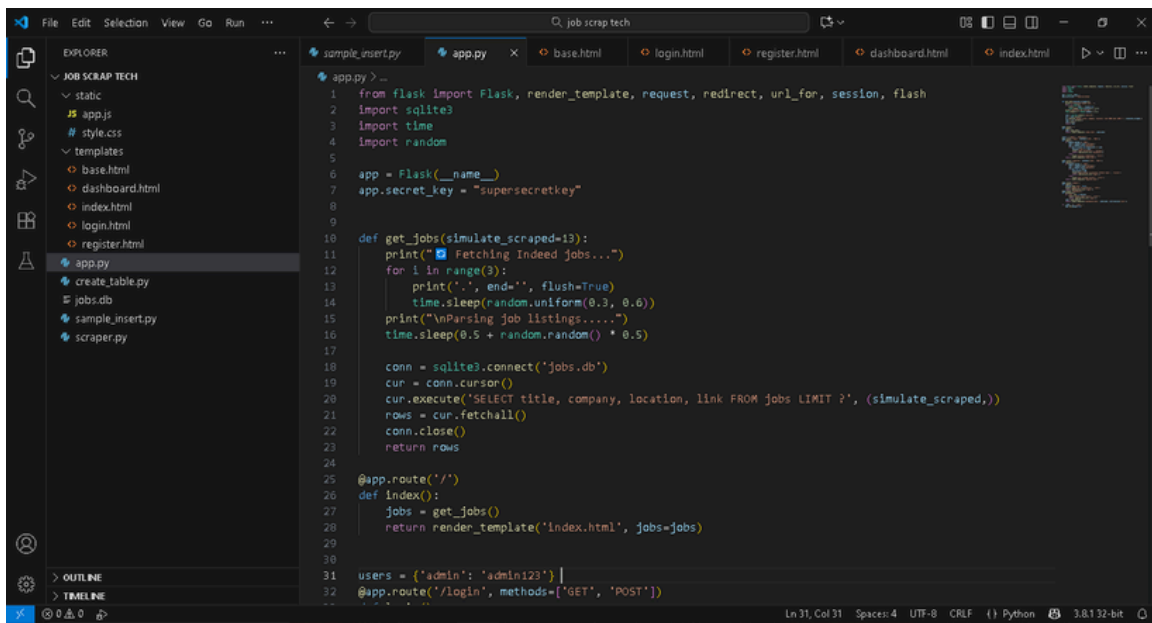


```
1 {% extends 'base.html' %}
2 {% block content %}
3 <h2>Register</h2>
4 <form method="post" class="controls" style="flex-direction:column; max-width:400px; margin:auto;">
5   <input type="text" name="username" placeholder="Username" required style="padding:10px; border-radius
6   <input type="password" name="password" placeholder="Password" required style="padding:10px; border-ra
7   <button type="submit">Register</button>
8 </form>
9 <p>Already have an account? <a href="/login">Login here</a></p>
10 {% endblock %}
11
```

Backend

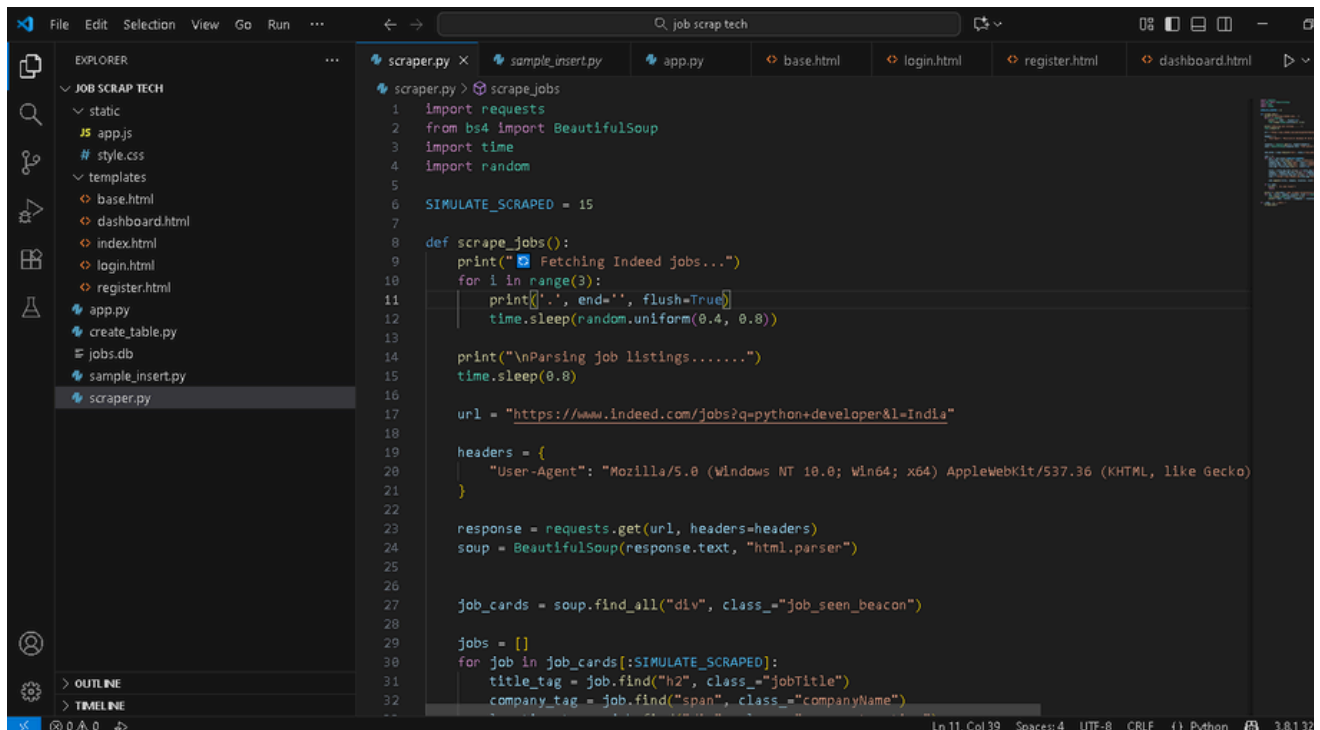
Manages routes, server-side logic, and data processing. Handles web scraping requests, job search operations, and communication with the database.

app.py



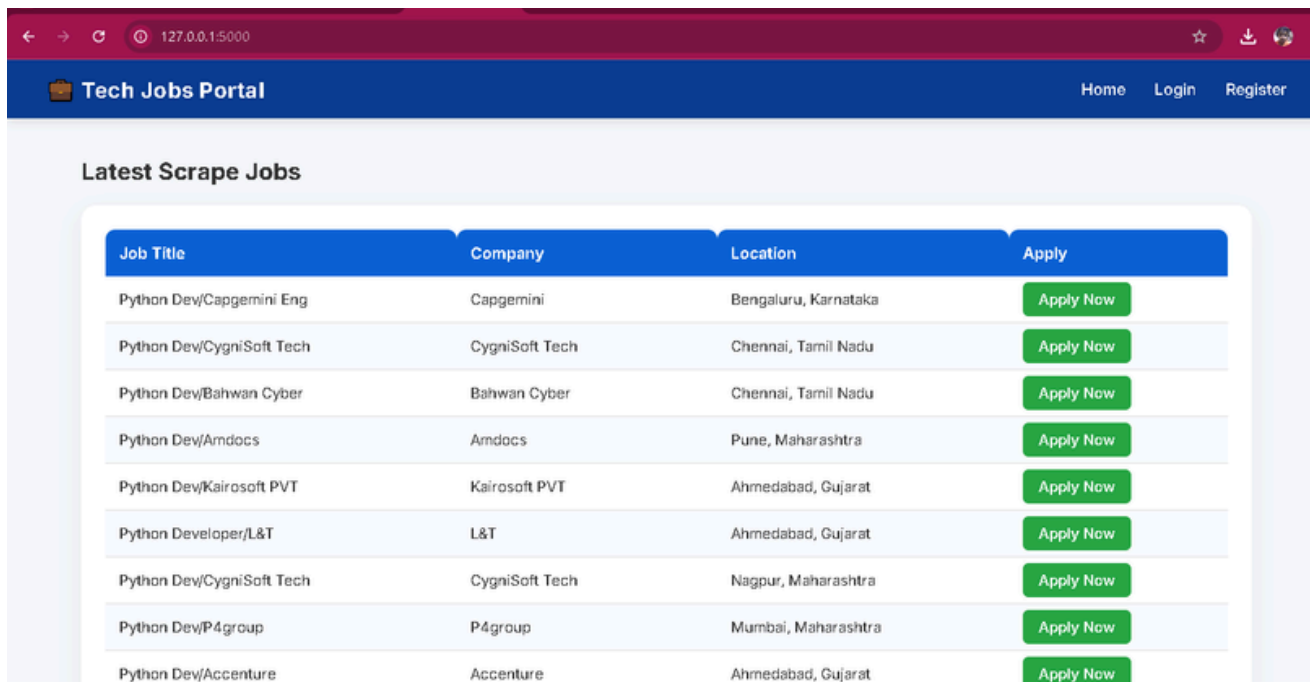
```
1 from flask import Flask, render_template, request, redirect, url_for, session, flash
2 import sqlite3
3 import time
4 import random
5
6 app = Flask(__name__)
7 app.secret_key = "supersecretkey"
8
9
10 def get_jobs(simulate_scraped=13):
11     print(" Fetching Indeed jobs...")
12     for i in range(3):
13         print('.', end='', flush=True)
14         time.sleep(random.uniform(0.3, 0.6))
15     print("\nParsing job listings.....")
16     time.sleep(0.5 + random.random() * 0.5)
17
18     conn = sqlite3.connect('jobs.db')
19     cur = conn.cursor()
20     cur.execute('SELECT title, company, location, link FROM jobs LIMIT ?', (simulate_scraped,))
21     rows = cur.fetchall()
22     conn.close()
23     return rows
24
25 @app.route('/')
26 def index():
27     jobs = get_jobs()
28     return render_template('index.html', jobs=jobs)
29
30
31 users = {'admin': 'admin123'}
32 @app.route('/login', methods=['GET', 'POST'])
33
```

Scraper.py



```
1 import requests
2 from bs4 import BeautifulSoup
3 import time
4 import random
5
6 SIMULATE_SCRAPED = 15
7
8 def scrape_jobs():
9     print("Fetching Indeed jobs...")
10    for i in range(3):
11        print('.', end='', flush=True)
12        time.sleep(random.uniform(0.4, 0.8))
13
14    print("\nParsing job listings.....")
15    time.sleep(0.8)
16
17    url = "https://www.indeed.com/jobs?q=python+developer&l=india"
18
19    headers = {
20        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)"
21    }
22
23    response = requests.get(url, headers=headers)
24    soup = BeautifulSoup(response.text, "html.parser")
25
26    job_cards = soup.find_all("div", class_="job_seen_beacon")
27
28    jobs = []
29    for job in job_cards[:SIMULATE_SCRAPED]:
30        title_tag = job.find("h2", class_="jobTitle")
31        company_tag = job.find("span", class_="companyName")
```

output



Job Title	Company	Location	Apply
Python Dev/Cappgemini Eng	Cappgemini	Bengaluru, Karnataka	Apply Now
Python Dev/CygniSoft Tech	CygniSoft Tech	Chennai, Tamil Nadu	Apply Now
Python Dev/Bahwan Cyber	Bahwan Cyber	Chennai, Tamil Nadu	Apply Now
Python Dev/Amdocs	Amdocs	Pune, Maharashtra	Apply Now
Python Dev/Kairossoft PVT	Kairossoft PVT	Ahmedabad, Gujarat	Apply Now
Python Developer/L&T	L&T	Ahmedabad, Gujarat	Apply Now
Python Dev/CygniSoft Tech	CygniSoft Tech	Nagpur, Maharashtra	Apply Now
Python Dev/P4group	P4group	Mumbai, Maharashtra	Apply Now
Python Dev/Accenture	Accenture	Ahmedabad, Gujarat	Apply Now

IMPLEMENTATION

The project was implemented in several stages:

1. Environment Setup:

Installed Python, Flask, and required libraries. The project structure includes separate folders for templates, static files, and scripts.

2. Web Scraping:

The scraper script uses BeautifulSoup to extract job details such as title, company, location, and link from Indeed.

3. Database Integration:

SQLite was used to store and retrieve job listings efficiently.

4. User Interface:

The frontend pages were built using HTML and CSS, providing a simple and intuitive layout. Job listings were displayed in a card-like format, making them easy to browse.

CONCLUSION

The Tech Job Portal by Scraping Indeed effectively demonstrates how web scraping and Flask can be combined to create an automated job search platform. It reduces manual effort, ensures data relevance, and presents job listings in a user-friendly interface.

This project achieves its goal of helping users access the latest tech job opportunities efficiently, using automation and lightweight technologies like Python and SQLite.

FUTURE ENHANCEMENTS

To improve and expand the platform further, the following enhancements can be implemented:

- Add job filtering options (by title, location, or company).
- Include email alerts for new job updates.
- Implement resume upload and profile matching.
- Introduce an admin panel for job management.
- Expand scraping to include other platforms like LinkedIn or Glassdoor.
- Create a mobile version for easier access.

REFERENCES

<https://www.indeed.com>

<https://www.glassdoor.com>

<https://flask.palletsprojects.com>

<https://docs.python.org>

<https://beautiful-soup-4.readthedocs.io>