INDEED SCRAPER

1. Abstract

This project focuses on the design and implementation of a lightweight **job portal web application** that centralizes job listings and allows users to search effectively. Built using the Flask framework in Python, the system reads data stored in JSON format, which was initially gathered through web scraping methods.

The application supports filtering by job title and location, helping users quickly narrow down opportunities that suit their interests. The frontend interface, developed with HTML and CSS, ensures a responsive and clean layout. Unlike static job boards, this portal dynamically updates search results, giving it an interactive and user-friendly experience.

This prototype highlights how simple technologies like Flask, JSON, and basic web design principles can be combined to build a functional and extendable job search tool.

2. Introduction

With the rapid expansion of online hiring, job seekers often face the challenge of navigating multiple platforms to find suitable positions. Having a unified portal simplifies this process by aggregating listings into one interface.

This project demonstrates how a job portal can be created using Flask as the backend framework. The system loads job postings from a JSON file generated via web scraping. Users can search for jobs based on keywords (e.g., “Python Developer,” “Backend Engineer”) and locations (e.g., “Remote,” “Bangalore”). Results are displayed in styled job cards that include the title, company, and location.

The objective is to showcase how structured data, combined with Flask’s routing and rendering capabilities, can be turned into a minimal but effective solution for job discovery.

3. System Requirements

Hardware Requirements

- Minimum 4 GB RAM (8 GB preferred for smoother scraping)
- Intel i3 processor or higher (i5 recommended)
- Around 500 MB free storage for files and dependencies

Software Requirements

- OS: Windows 10/Linux/Mac
- Python 3.8 or higher
- Flask framework for backend
- Libraries: JSON (data handling), Jinja2 (template rendering)
- Tools: Any IDE (VS Code, PyCharm), modern browser (Chrome/Firefox)
- Package Manager: pip for installations

These requirements make the project easily deployable on any modern laptop or PC.

4. Project Architecture

The project is structured on a **client–server model** using Flask as the backend server.

- **Data Layer:** Job postings are stored in a jobs.json file.
- **Application Layer (Flask):** Handles requests, processes queries, and filters results.
- **Presentation Layer (HTML + CSS):** Displays jobs in a responsive grid card layout.

Execution Flow:

1. User opens the portal homepage.
2. A search query (job title/location) is entered.
3. Flask loads data from jobs.json.
4. Results are filtered and passed to Jinja2 templates.
5. The browser displays results as job cards.

This modular design allows for scalability and easy future upgrades.

5. Implementation

Folder Structure:

```
job-portal/
├── app.py
├── jobs.json
├── templates/
│   └── jobs-1.html
├── static/
│   ├── style.css
│   └── logo.png
```

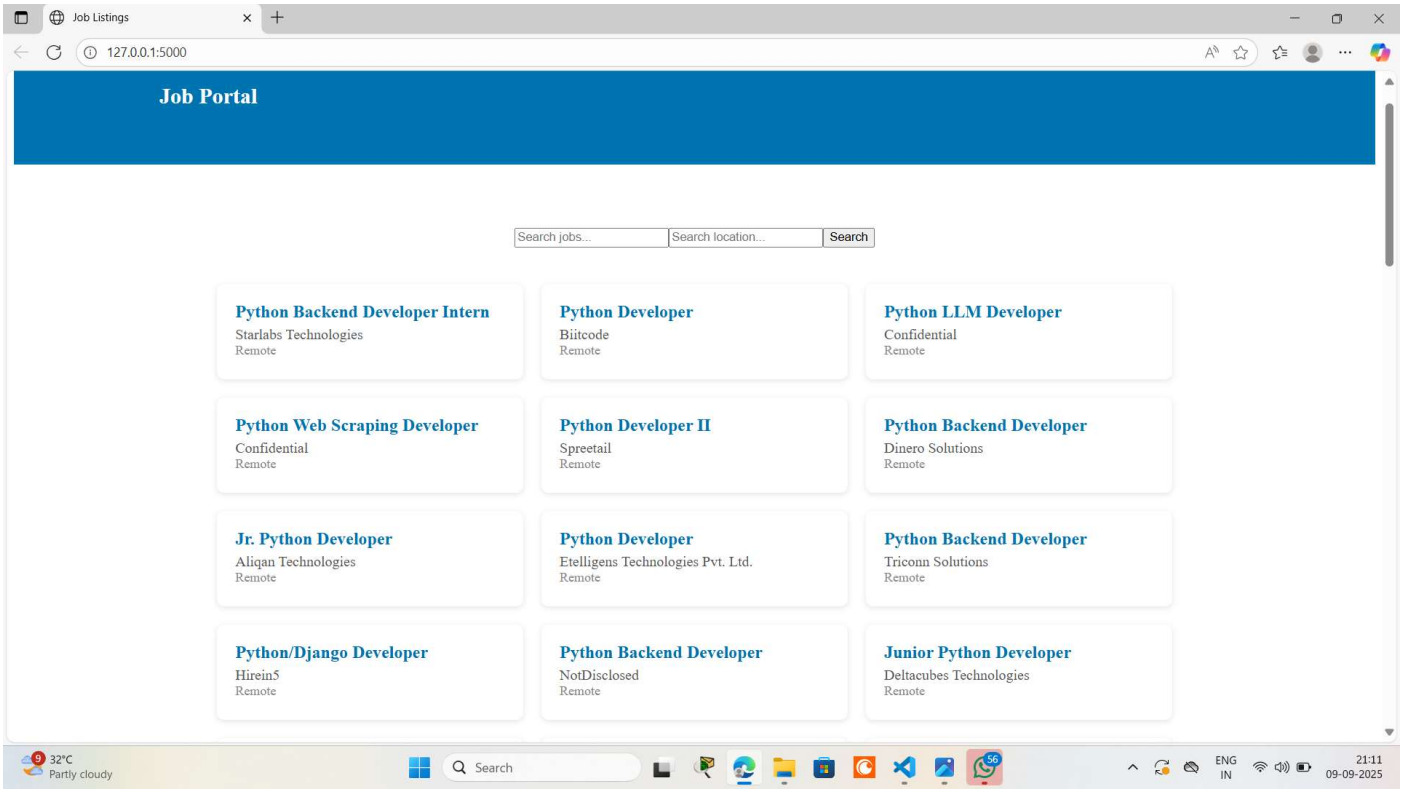
Backend (Flask - app.py):

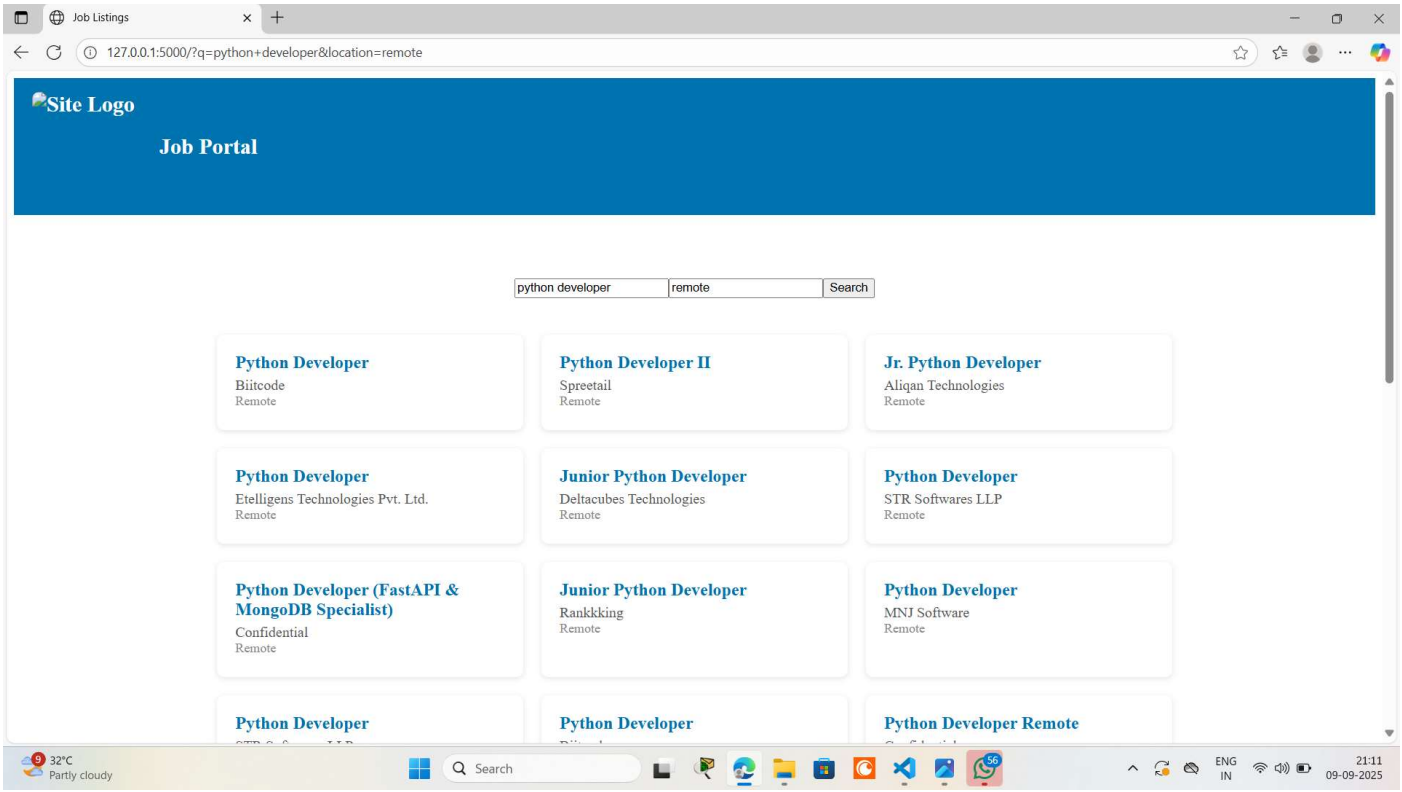
- Loads data from JSON file
- Processes user queries
- Filters jobs by title/location
- Sends filtered results to templates

Frontend (HTML + CSS):

- Search bar with job title & location fields
- Responsive grid-based job cards
- Styling for hover effects and mobile compatibility

6. Output


Job Listings
127.0.0.1:5000
Job Portal
Search jobs...
Search location...
Search
Python Backend Developer Intern
Starlabs Technologies
Remote
Python Developer
Biitcode
Remote
Python LLM Developer
Confidential
Remote
Python Web Scraping Developer
Confidential
Remote
Python Developer II
Spreetail
Remote
Python Backend Developer
Dinero Solutions
Remote
Jr. Python Developer
Aliqan Technologies
Remote
Python Developer
Etelligens Technologies Pvt. Ltd.
Remote
Python Backend Developer
Triconn Solutions
Remote
Python/Django Developer
Hirein5
Remote
Python Backend Developer
NotDisclosed
Remote
Junior Python Developer
Deltacubes Technologies
Remote
32°C
Partly cloudy
Search
ENG
IN
21:11
09-09-2025


Job Listings
127.0.0.1:5000/?q=python+developer&location=remote
Site Logo
Job Portal
python developer
remote
Search
Python Developer
Biitcode
Remote
Python Developer II
Spreetail
Remote
Jr. Python Developer
Aliqan Technologies
Remote
Python Developer
Etelligens Technologies Pvt. Ltd.
Remote
Junior Python Developer
Deltacubes Technologies
Remote
Python Developer
STR Softwares LLP
Remote
Python Developer (FastAPI & MongoDB Specialist)
Confidential
Remote
Junior Python Developer
Rankkking
Remote
Python Developer
MNJ Software
Remote
Python Developer
Python Developer
Python Developer Remote
32°C
Partly cloudy
Search
ENG
IN
21:11
09-09-2025

7. Testing

Testing was conducted using the following scenarios:

- **Keyword Test:** Searching “Python” returns all jobs containing “Python.”
- **Location Test:** Searching “Remote” filters all remote jobs.
- **Combined Test:** Searching “Python” + “Remote” narrows results further.
- **Invalid Input Test:** Searching “Doctor” returns no results.
- **Case Sensitivity Test:** Queries like “PYTHON” still match correctly.

All cases worked as expected.

8. Advantages and Limitations

Advantages

- Easy to run locally, minimal setup
- JSON-based storage makes updating jobs simple
- Search supports both title and location
- Responsive design works across devices

Limitations

- Dataset is static; no live scraping yet
- Lacks advanced filters (salary, job type, experience)
- No authentication or user profiles
- Requires manual JSON updates

9. Future Enhancements

- Add real-time scraping with BeautifulSoup or Selenium
- Store jobs in a database (MySQL/MongoDB)
- Enable user accounts for job saving and alerts
- Support advanced filtering (skills, salary, experience)
- Integrate APIs from platforms like Indeed or LinkedIn

10. Conclusion

The project successfully demonstrates how a job search portal can be built with Flask, JSON, and a simple UI. Although it currently runs on static data, the system forms a solid foundation for future improvements such as real-time scraping, authentication, and advanced filtering.

With additional features, this prototype can evolve into a full-fledged job portal that significantly improves the job-seeking experience.

Completed By,

Sathvikha V

Rajalakshmi Engineering College