**Tech Job Portal** by Scraping Indeed

**-L.Mahalakshmi**

**Batch1-Team4**

**Cybernaut intern**

**1. Problem Statement**

Job seekers often find it tedious to manually search and compile job listings across platforms like Indeed. Despite the availability of job portals, there is no automated, focused way to scrape tech job listings for specific domains in real-time and export the results for analysis.

**Goal**:  
Develop a web-based solution that allows users to select a tech domain, scrape real-time job data from **Indeed India**, and **view + download** results in a structured format.

**2. Objectives**

✅ Create a web interface to select predefined tech job domains.

✅ Use Selenium to automate scraping of job listings from **Indeed**.

✅ Parse job information including title, company, location, salary, post date, and link.

✅ Display scraped data in a well-formatted HTML table.

✅ Allow users to **download the data as a CSV file**.

✅ Handle dynamic web content and bot protection scenarios.

**3. Code Explanation**

**➤ app.py (Flask Application)**

Main backend file responsible for:

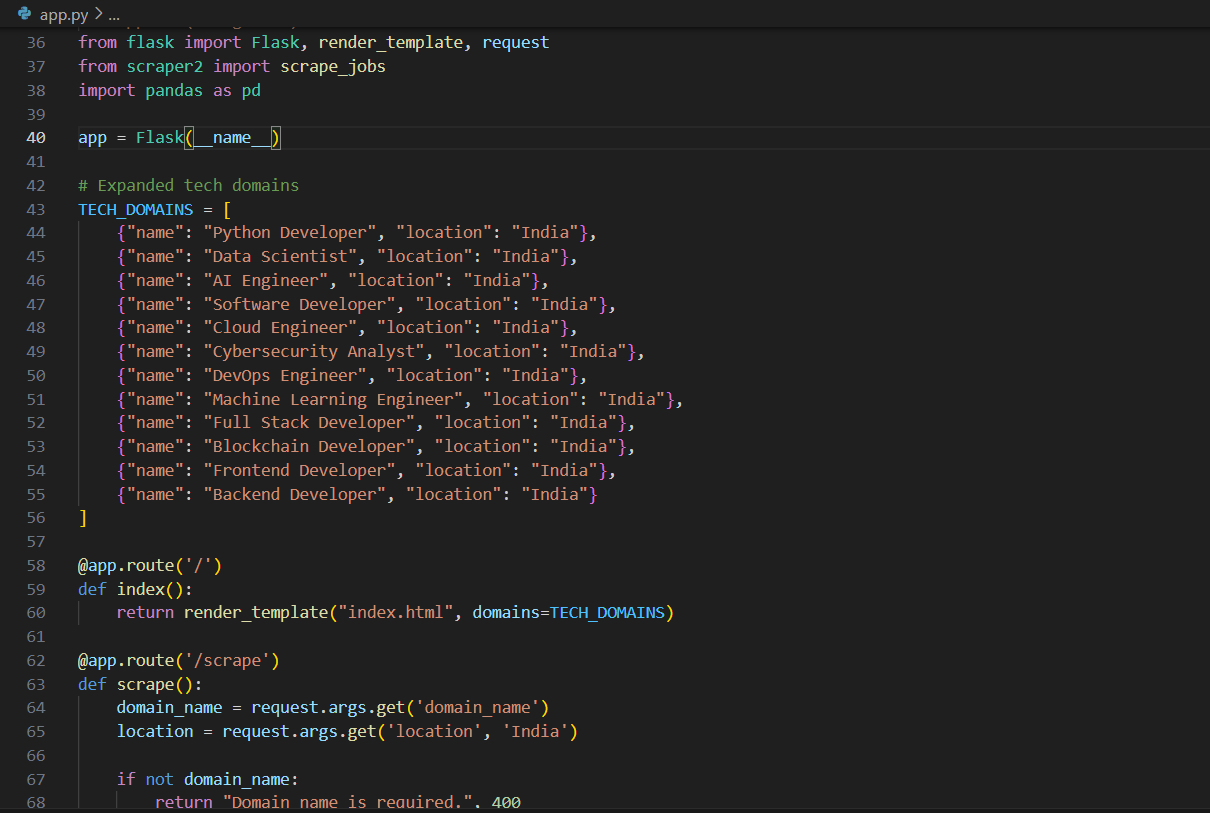
Routing web requests.

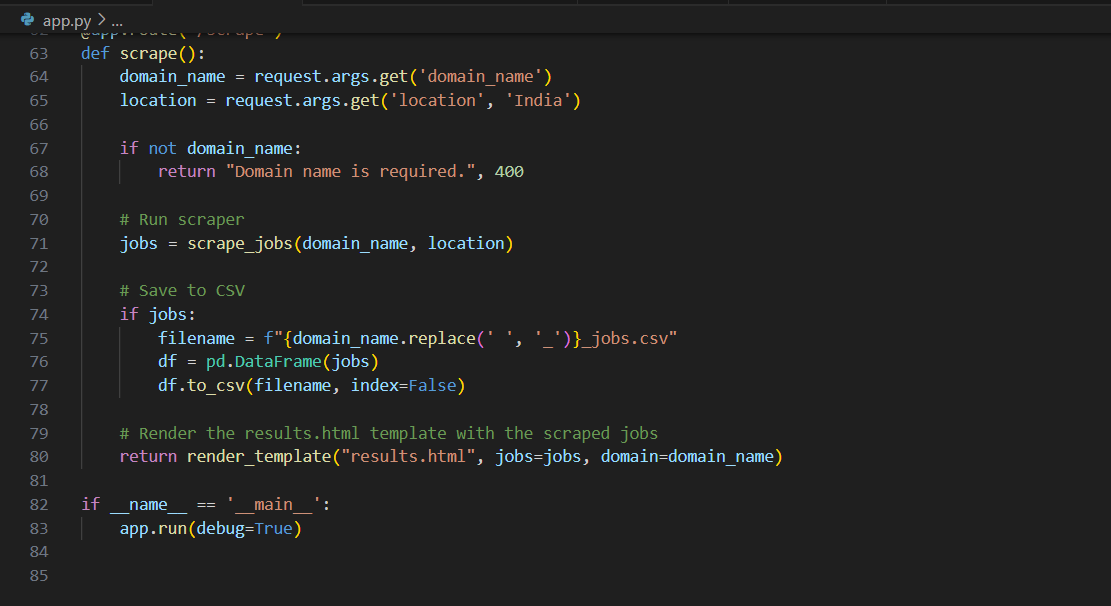
Interacting with the job scraper.

Rendering web pages.

Handles the web server logic and routing:

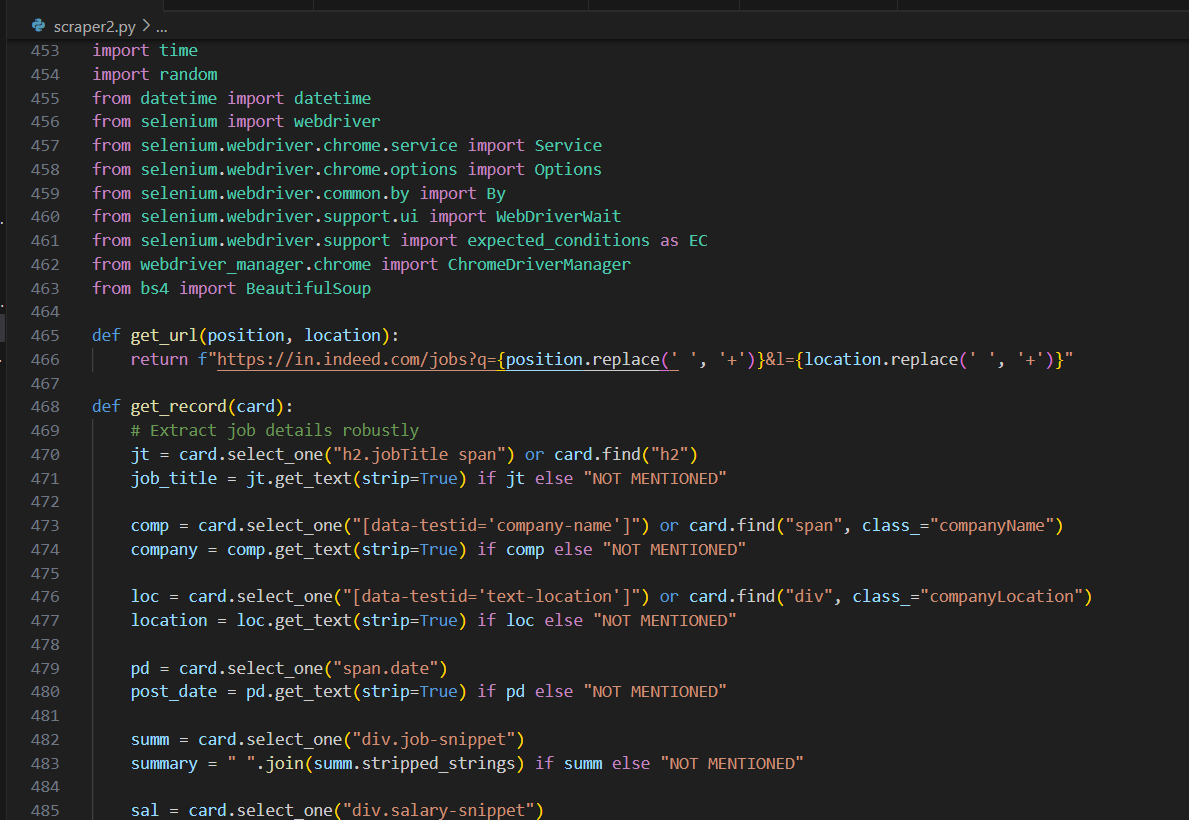
* **Route /**:
  + Renders index.html with a list of tech domains.
  + Users click on a domain to trigger scraping.

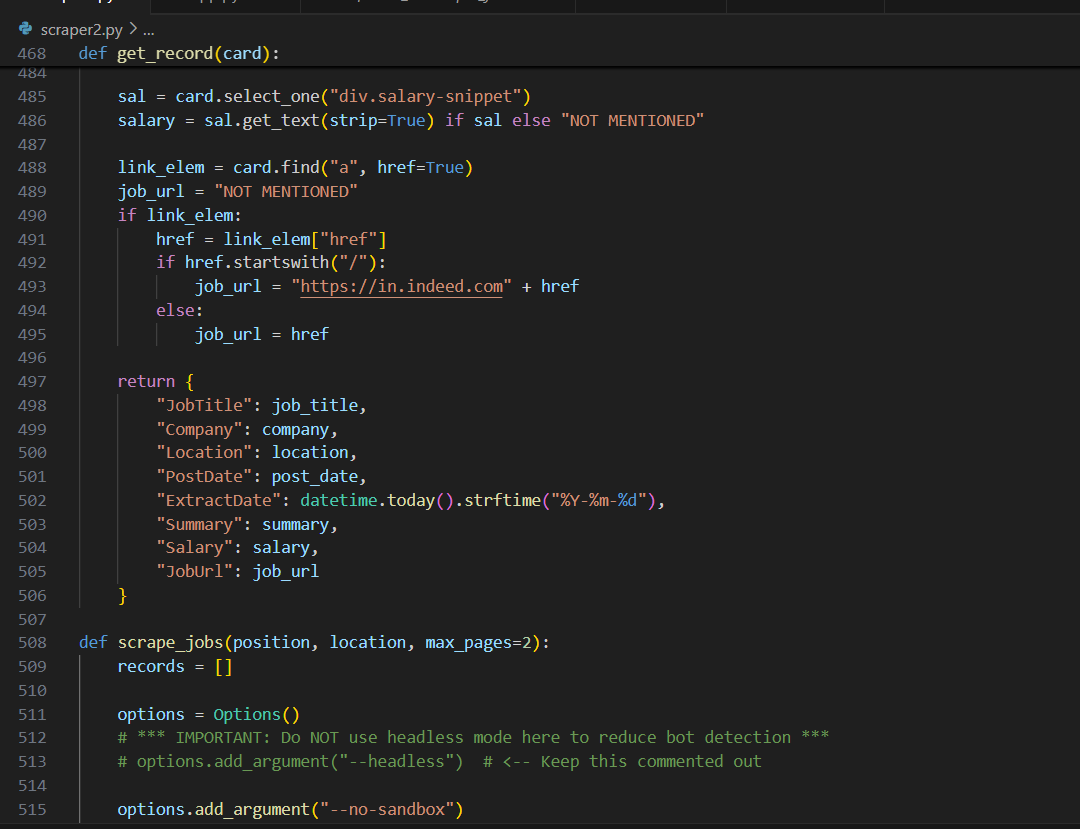


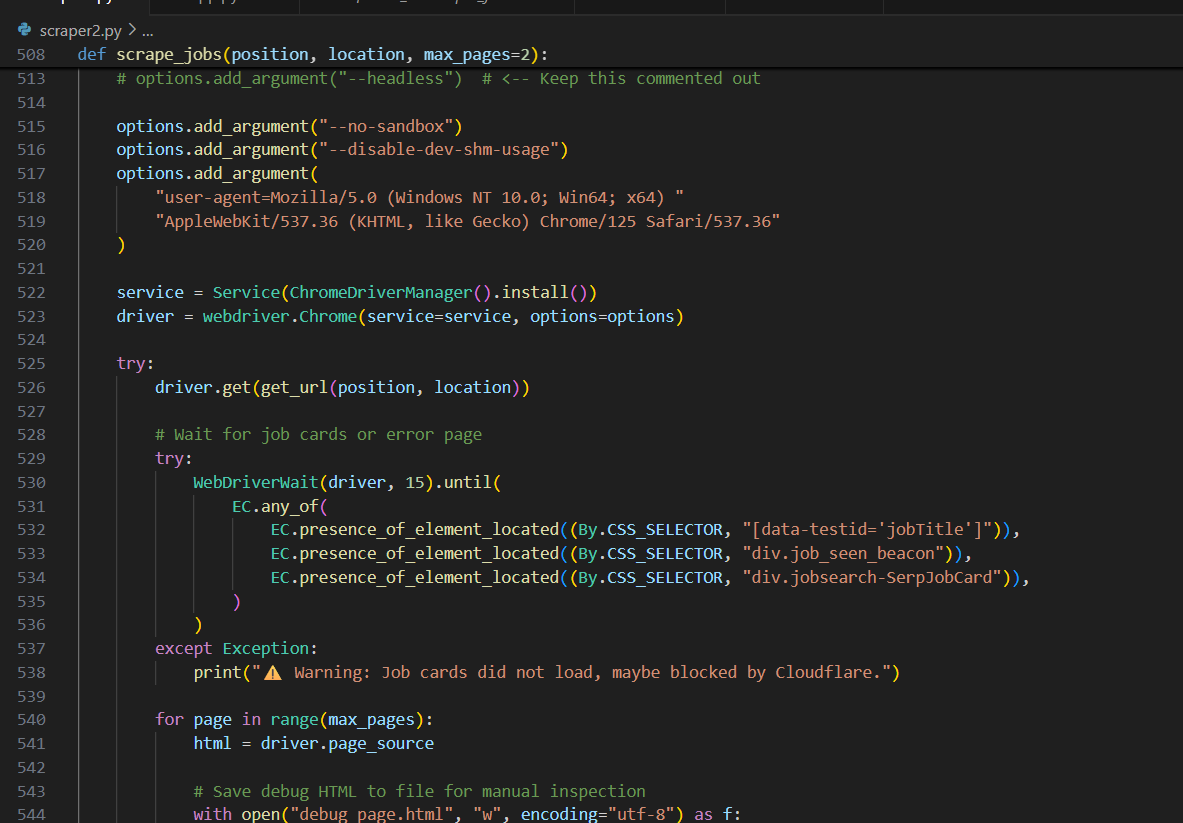
* **Route /scrape**:
  + Receives selected domain and location via GET.
  + Calls scrape\_jobs() from scraper2.py.
  + Converts results to a Pandas DataFrame and saves as CSV in static/
  + Passes job data to results.html for display and Saved as a .csv file locally.

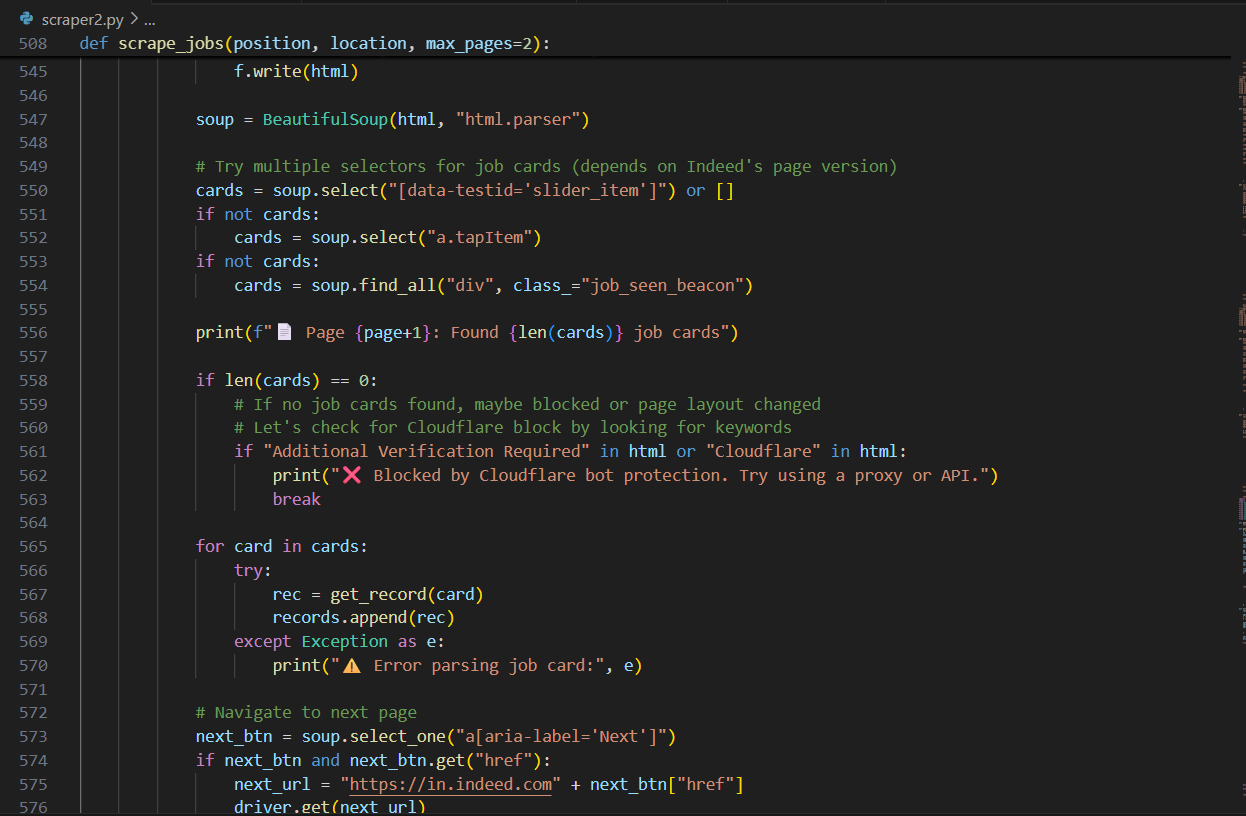
**➤ scraper2.py (Web Scraper)**

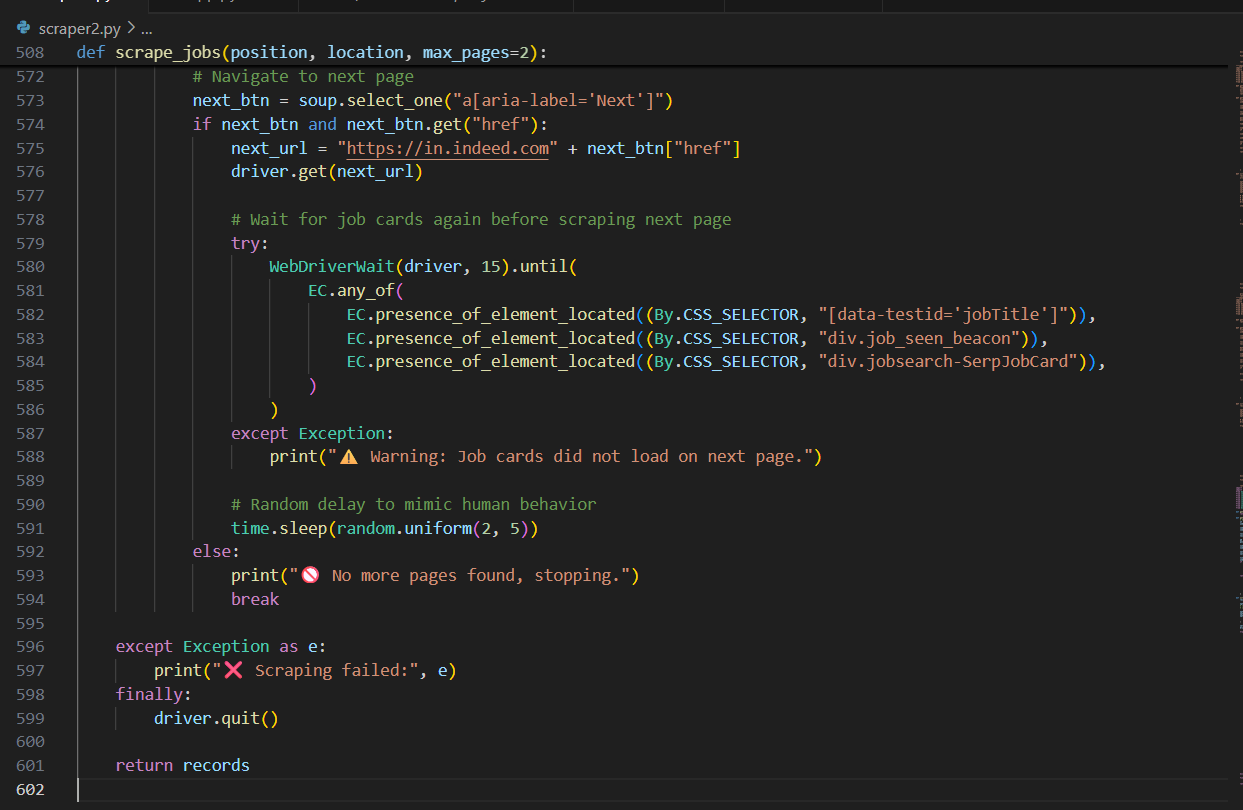
Automates Indeed.com job search using Selenium + BeautifulSoup:

* **get\_url(position, location)**: Builds Indeed search URL.
* **get\_record(card)**: Parses a single job card into a dictionary of:
  + JobTitle, Company, Location, PostDate, Salary, Summary, JobUrl
* **scrape\_jobs(position, location, max\_pages=2)**:
  + Launches Chrome using Selenium.
  + Navigates Indeed, waits for job elements.
  + Loops through 2 pages of results.
  + Extracts job cards using multiple CSS selectors for robustness.
  + Handles pagination and potential bot detection.
  + Returns a list of job dictionaries.





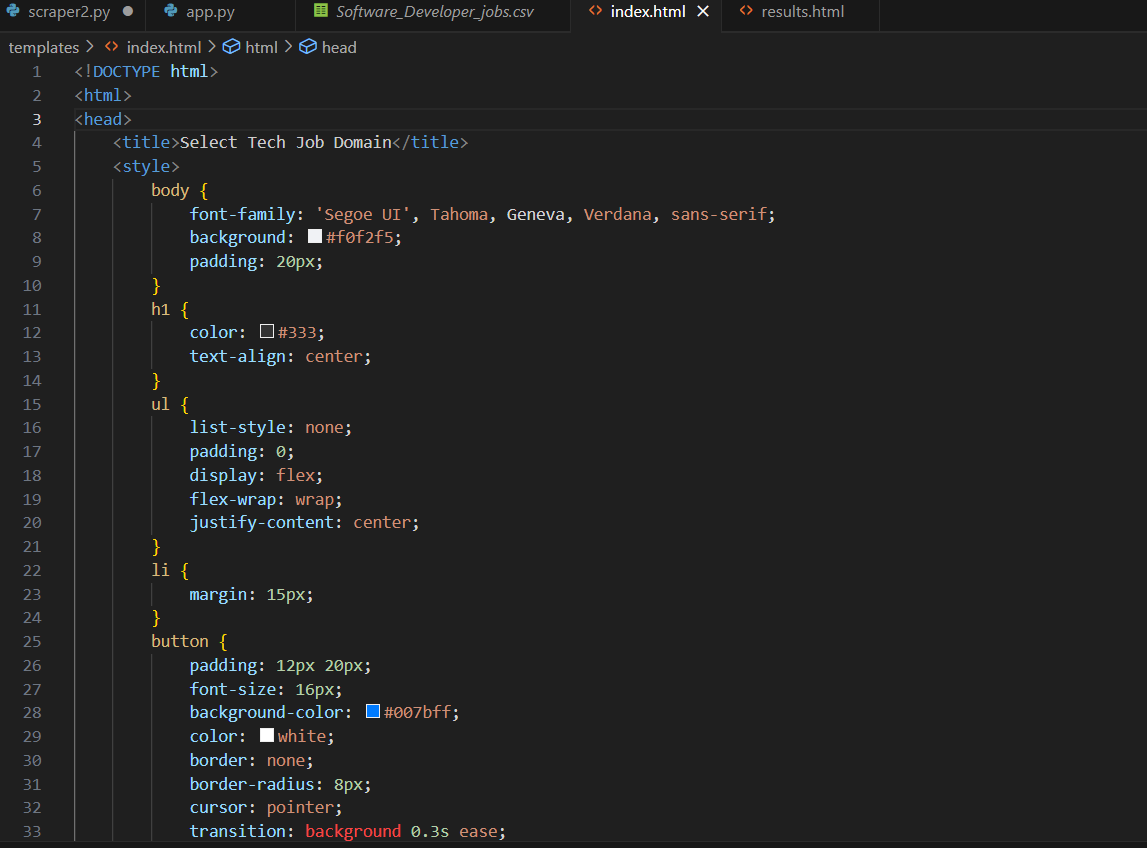


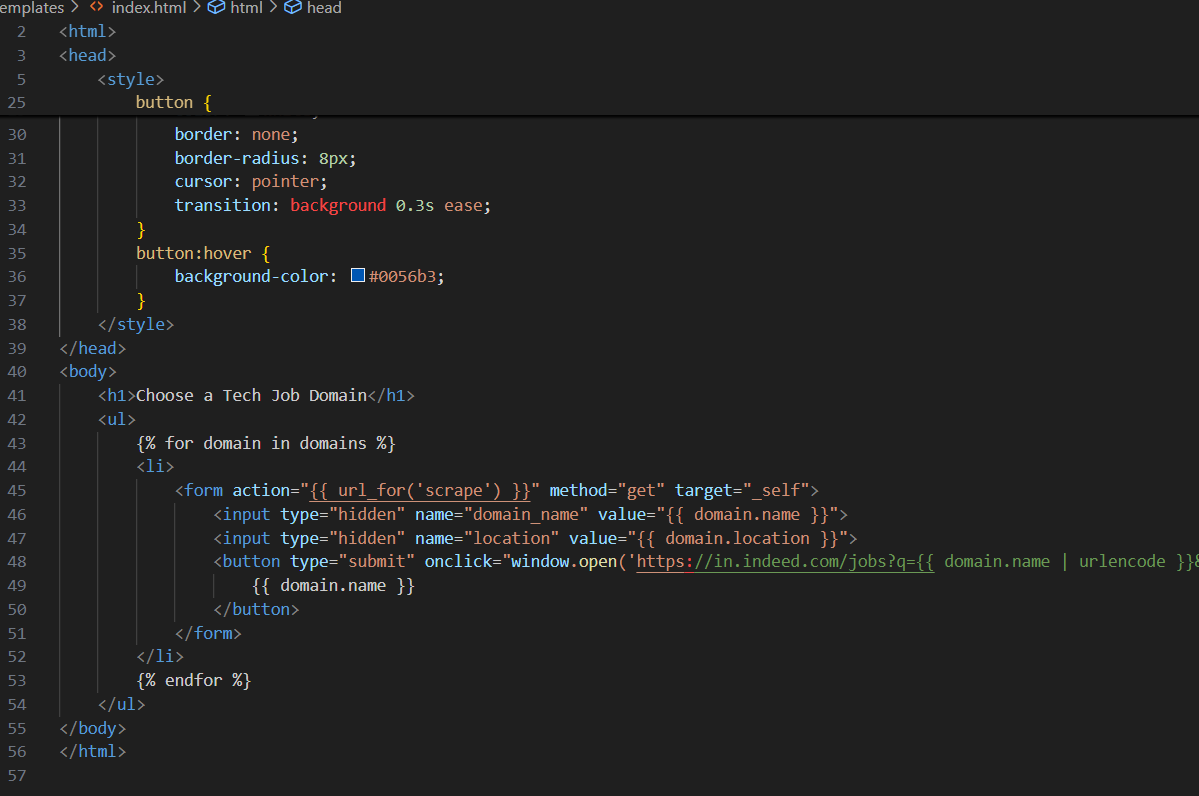


**➤ index.html**

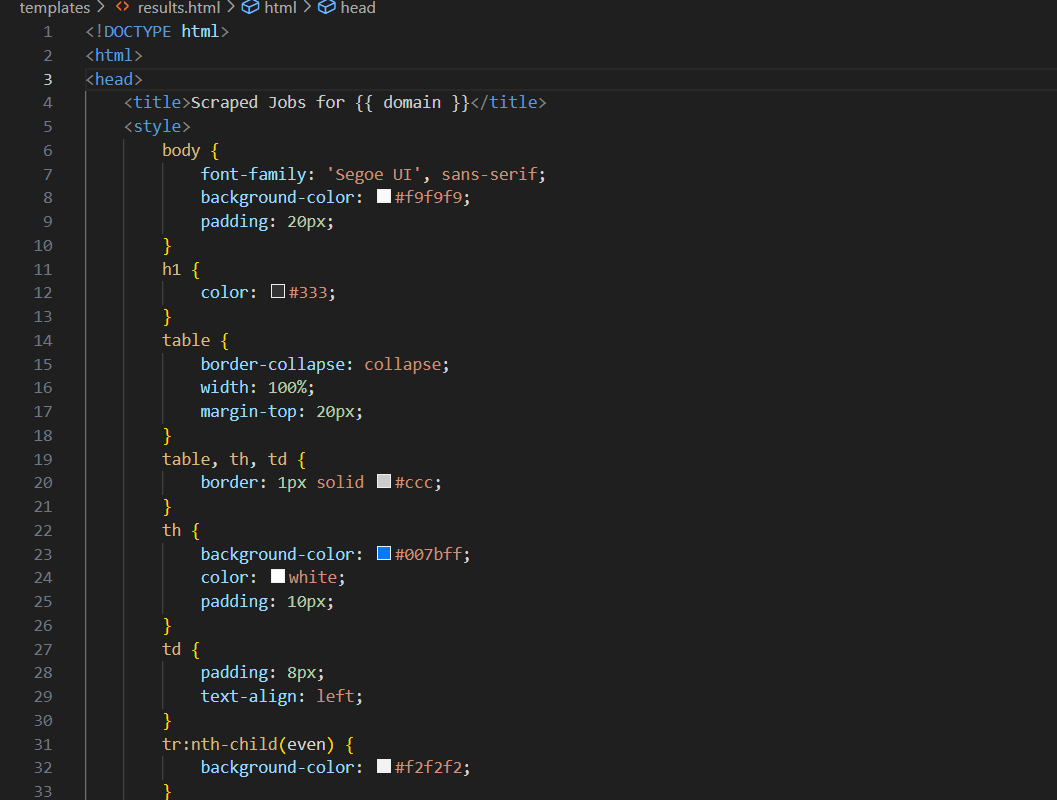
HTML template rendered at /.

* Lists all predefined job domains as buttons.
* Each button submits a form that triggers the /scrape route with domain and location.
* Highlights:
* Simple and user-friendly interface.
* Sends selected job title and location to Flask using a GET request.

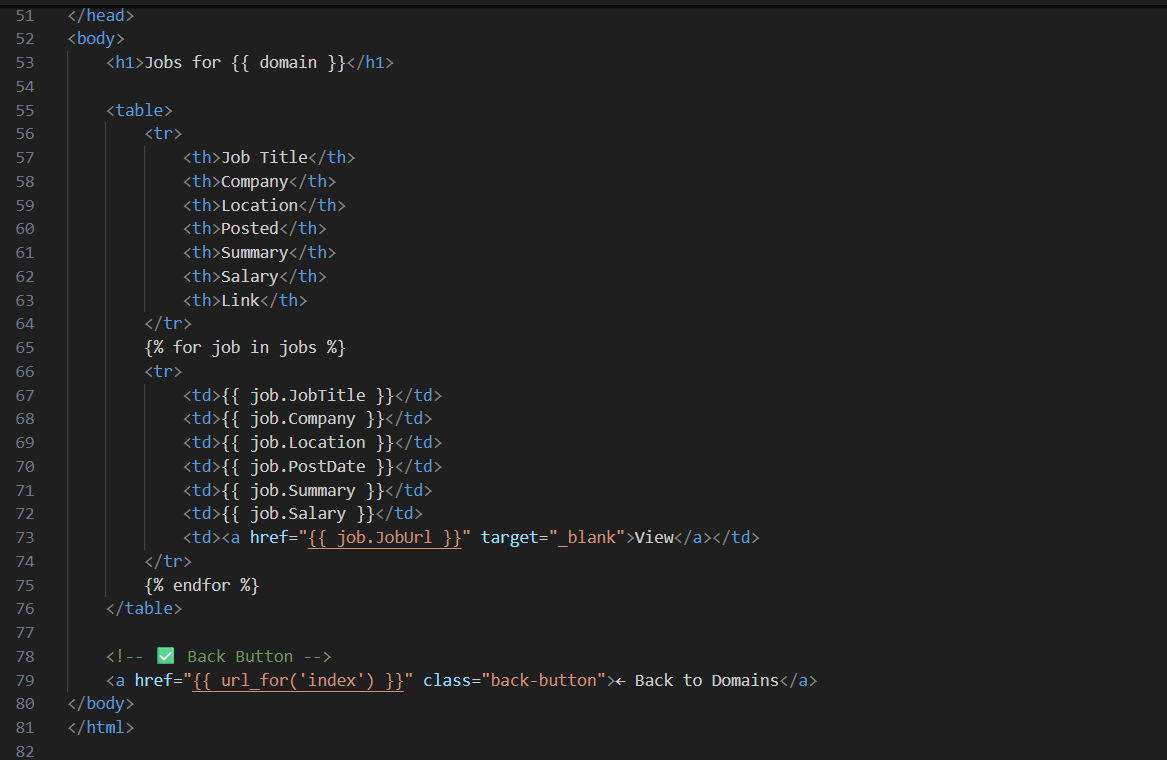




**➤ results.html**

* Displays job listings in a responsive table.
* Offers a **CSV download link**.
* Includes a **back button** to return to domain selection.





**4. Summary of Workflow**

 **User visits** the home page → chooses a job domain.

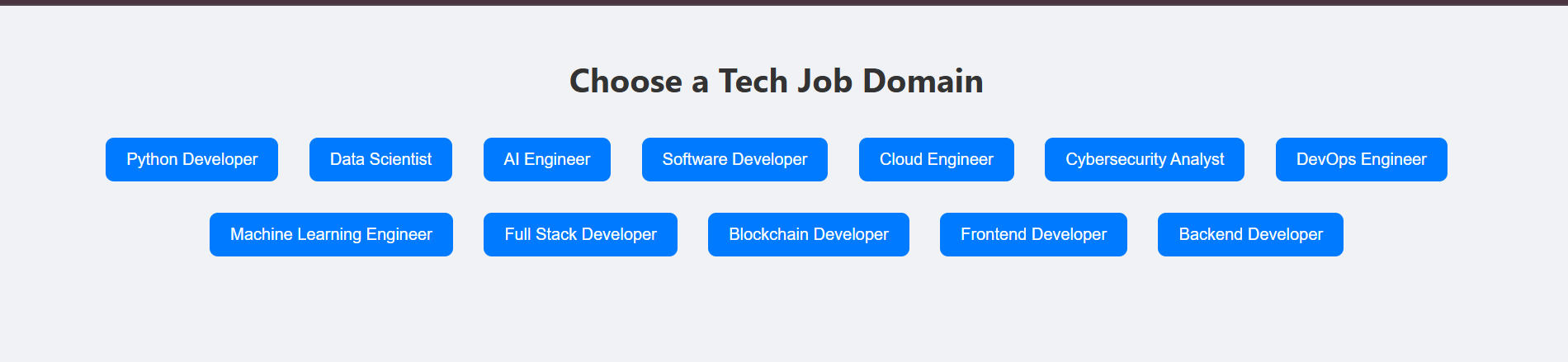
 Flask **sends domain info** to the scraper.

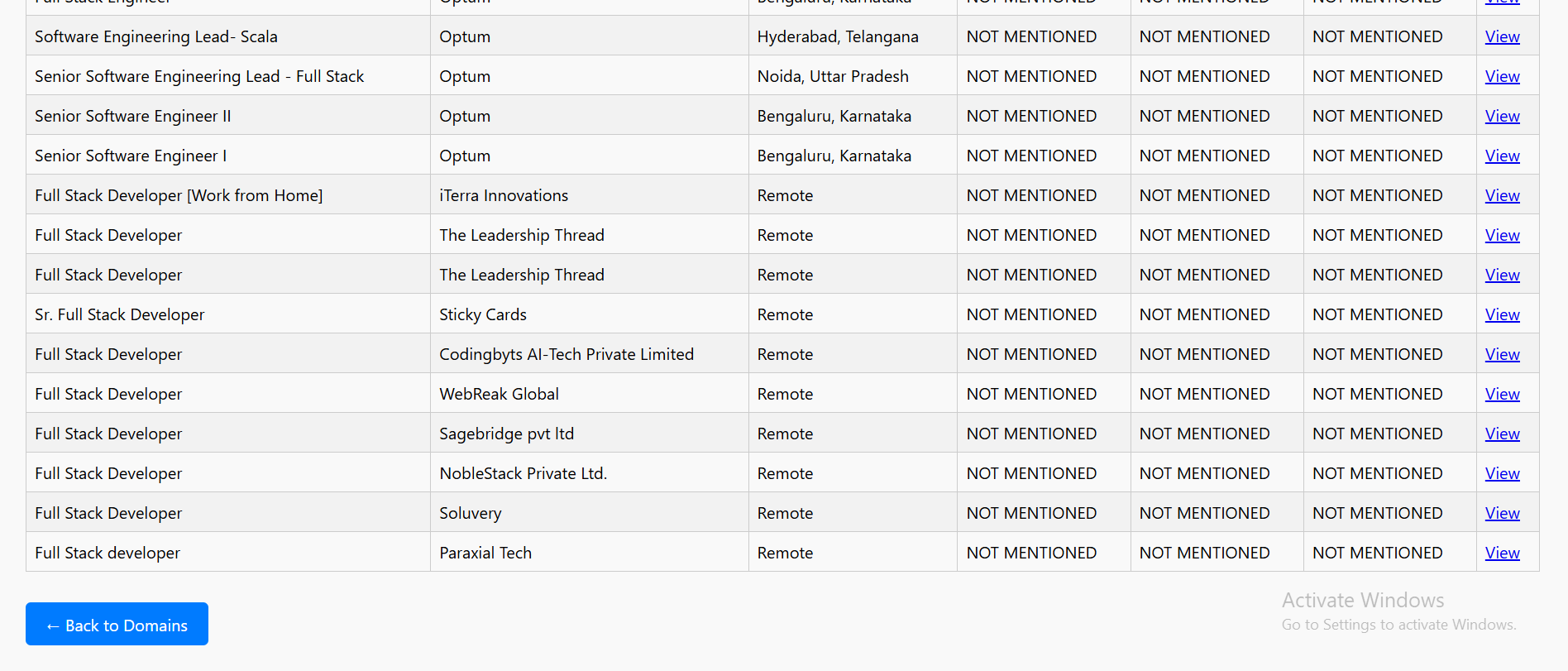
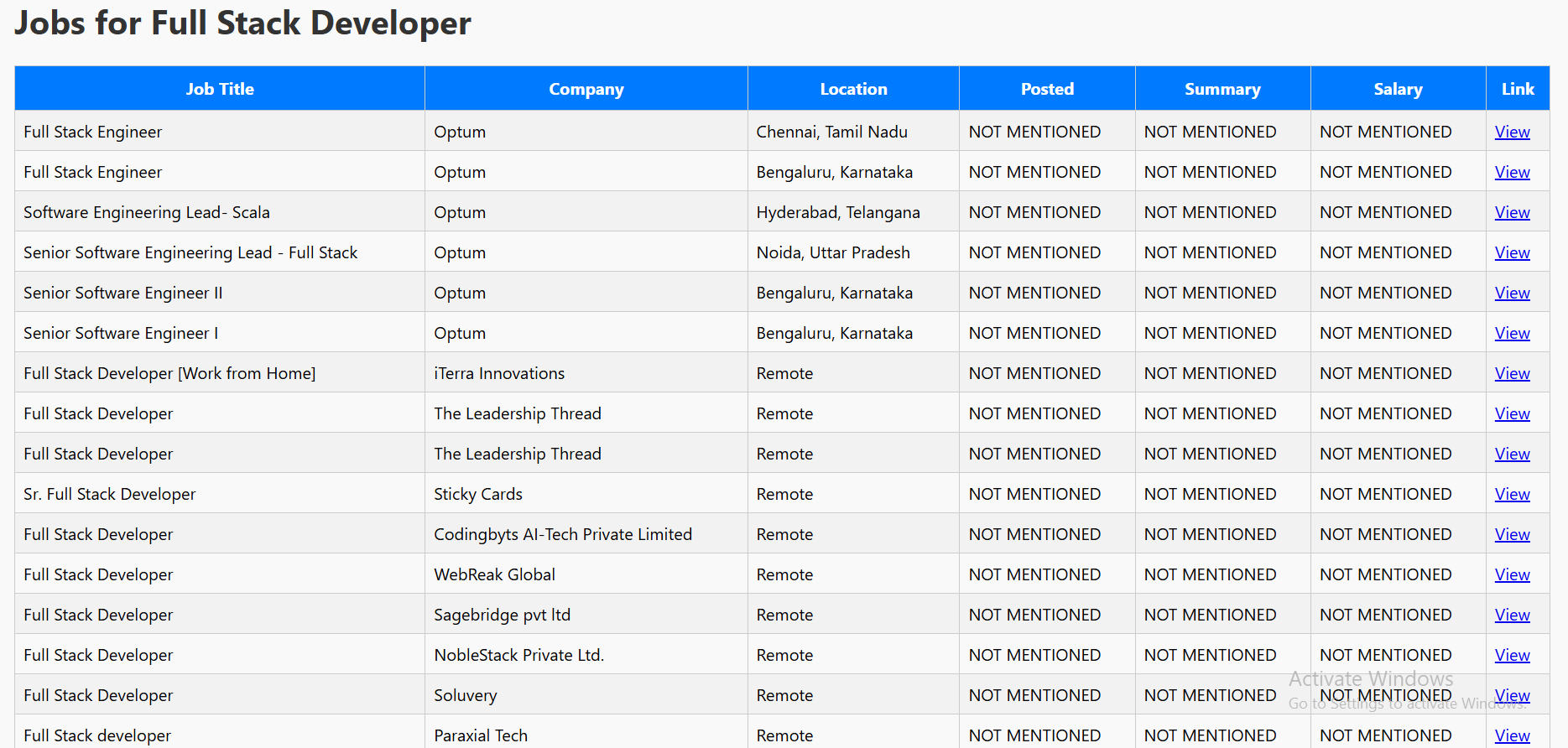
 **Selenium opens Indeed**, scrapes multiple job pages.

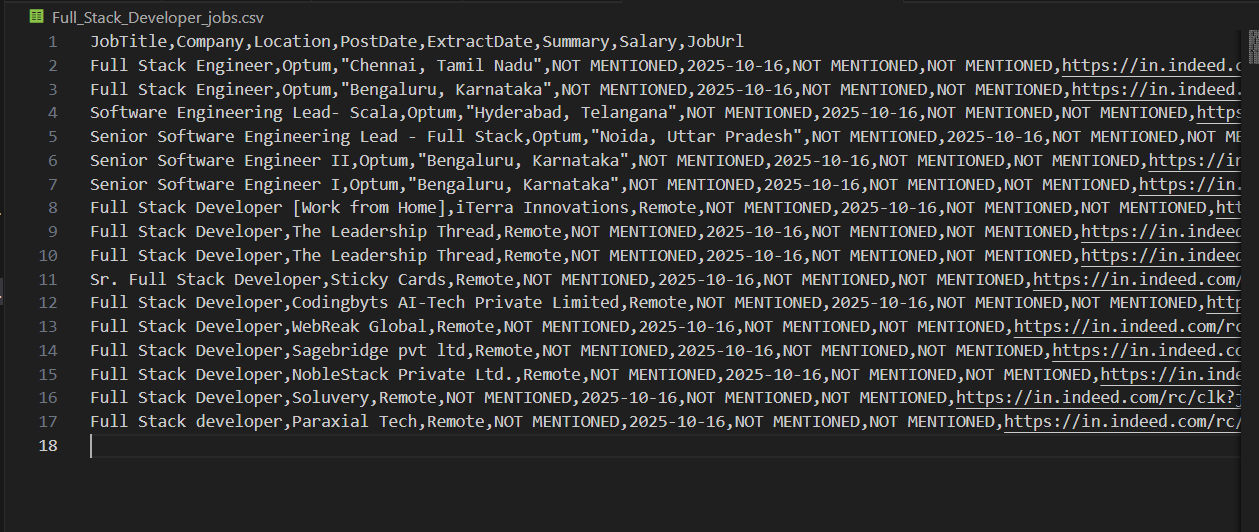
 Data is **parsed and saved** to a list and optionally CSV.

 **Jobs are displayed** on the results page.

 User can go **back to select** another domain.





****

**5. Issues Faced and How They Were Handled**

| **Issue** | **Description** | **Solution** |
| --- | --- | --- |
| **Bot Detection by Cloudflare** | Indeed uses Cloudflare to block scrapers. | Avoided headless mode and used user-agent headers. Future fix: proxies or APIs. |
| **Dynamic Page Structure** | Job card HTML structure varies. | Used multiple CSS selectors as fallbacks. |
| **Selenium Load Delays** | Pages took time to load fully. | Used WebDriverWait to wait for specific job elements. |
| **Duplicate Jobs / Empty Cards** | Not all cards were complete or valid. | Used exception handling around get\_record() to skip broken cards. |
| **CSV File Access** | Users couldn’t download CSV initially. | Saved files to Flask’s static/ directory and provided a direct download link. |
| **UI Confusion with Button Behavior** | onclick conflicted with form submission. | Simplified logic: removed JavaScript and kept standard form behavior. |

**Conclusion**

This project demonstrates how web scraping and web development can be combined to build a functional job listing tool. By using **Selenium** for dynamic content scraping and **Flask** for the user interface, the app allows users to easily search and view tech job listings from **Indeed India** in real time.