

Data Scraping Engineer – Trial Test

Overview

This trial test is designed to evaluate your practical skills in web scraping, data extraction, and code organization. The goal is to assess how you approach real-world scraping problems, handle errors, and produce clean, usable data.

Please read the instructions carefully and follow the requirements as closely as possible.

Task Description

Create a Python script that scrapes data from the following website:

Target URL: <https://scraping-trial-test.vercel.app>

Your script should extract the following fields from each record:

- Business Name
- Registration ID / Entity Number
- Status
- Filing Date
- Agent Details

If the site contains multiple pages or paginated results, your script should process **all available pages**.

Functional Requirements

Your solution must:

1. Use Python 3.x
2. Use one of the following approaches:
 - `requests + BeautifulSoup`, or
 - `Selenium / Playwright` for dynamic pages or any
3. Handle pagination or multiple result pages correctly
4. Save the final output to either:
 - `output.json`, or
 - `output.csv`
5. Include basic error handling for:

- Network errors
 - Missing or unexpected HTML elements
6. Include logging of errors or important events to a log file
-

Output Format

The output file should contain one record per business entity, for example:

```
{  
  "business_name": "ABC COMPANY LLC",  
  "registration_id": "123456",  
  "status": "Active",  
  "filing_date": "2023-05-14",  
  "agent_name": "Sara Davis",  
  "agent_address": "699 Broadway Ave",  
  "agent_email": "sara.davis.e71f523a99c3@example.com"  
}
```

If using CSV, make sure the header row clearly labels each column.

Deliverables

Please submit the following:

1. A GitHub repository containing your solution (public or private with access granted)
2. The Python script(s)
3. The generated output file (output.json or output.csv)
4. A short README.md file that explains:
 - How to install dependencies
 - How to run the script
 - Which libraries you used and why
 - Any assumptions or limitations

Repository requirements:

- Initialize a new repository for this task
- Commit your work with clear, logical commit messages
- Include a .gitignore file for Python projects
- Do not include virtual environments or large generated files

You may submit the repository link (preferred) or a ZIP file of the repository.

Optional Bonus (Not Required)

These are optional and will be considered a plus:

- Resume capability (script can continue from where it stopped)
 - Retry logic on temporary failures
 - Rate limiting / polite delays between requests
 - Separation of scraping and parsing logic
-

You do not need to over-engineer the solution. We are more interested in:

- Code clarity and structure
 - Correctness of extracted data
 - Thoughtful error handling
 - Clean, readable implementation
-

Evaluation Criteria

Your submission will be evaluated based on:

- Correctness of extracted fields
 - Ability to handle pagination
 - Code readability and structure
 - Proper error handling and logging
 - Quality of documentation in the README
-

Good luck, and we look forward to reviewing your solution.