

Deck | Technical Assignment

Web Scraping with Playwright

Aim

This project assesses the candidate's ability to use Playwright for web scraping, handling authentication, and data extraction. The task involves creating a Python script that logs into a target website and extracts specific data, simulating real-world scenarios.

How do we assess quality?

Poor generic, non-actionable for Deck.

OK somewhat actionable, but uninspiring.

Impressive heavily insightful and actionable by Deck.

Evaluation Criteria

Correctness Does the script accomplish the task?

Code Quality Is the code clean, well-structured, and maintainable? **Testing:** Are thorough tests provided to validate functionality?

Problem Solving: Handling of potential challenges like dynamic content or failures?

Instructions

Provide your work to marc@deck.co & keith@deck.co by sending a URL to a GitHub Repo before Feb 3, 2025.

The project is designed to be completed within *2 hours*, with each step building on the previous one.

Depending on the quality of the work, we will determine whether we will proceed with the Technical Interview.

Hints

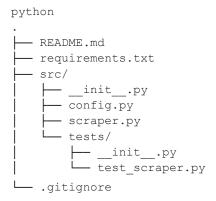
- 1. We Encourage the use of environment variables or a config file for sensitive data.
- 2. Ensure proper error handling is in place for robustness.

Target Website

https://deck-dev-eastus2-academy.yellowrock-2749f805.eastus2.azurecontainerapps.io/



Project Structure



Step-by-Step Instructions

1. *Setup the Project:*

- a. Install the required dependencies using pip install -r requirements.txt.
- b. Ensure you have python and Playwright installed.

2. *Understand the Target Site:*

- a. Navigate to the provided URL.
- b. Analyze how to perform a login using the Advanced 2FA

3. *Basic Authentication:*

- a. Implement a function in scraper.py to automate the login process.
- b. Use Playwright to navigate to the login page, fill in the credentials, and submit the form.
- c. *Test Case*: Write a test in test_scraper.py to check if login is successful by verifying page content after login.

4. *Data Extraction:*

- a. Identify specific data elements on the page after login (e.g., user info, dashboards, etc.).
- Extract this data and store it in a structured format (e.g., JSON). Specifically the account information, address, due dates and last months usage
- c. Implement error handling for common issues like elements not found or network errors.
- d. Download Latest Bill for each address // download all Recent Statements.



5. *Data Storage:*

- a. Write a function to save the extracted data to a JSON file within the project directory.
- b. *Test Case*: Ensure that the JSON file is correctly created and contains the expected data structure.

6. *Advanced Features:*

- a. Handle pagination if data spans multiple pages.
- b. Implement a delay between requests to avoid being rate-limited.

7. *Testing Framework:*

- a. Use pytest for test-driven development.
- b. Run tests using ${\tt pytest}\ {\tt tests}/$ to ensure all functionality works as expected.

** END OF ASSIGNMENT **