

# 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?

<b>Poor</b>	generic, non-actionable for Deck.
<b>OK</b>	somewhat actionable, but uninspiring.
<b>Impressive</b>	heavily insightful and actionable by Deck.

### Evaluation Criteria

<b>Correctness</b>	Does the script accomplish the task?
<b>Code Quality</b>	Is the code clean, well-structured, and maintainable?
<b>Testing:</b>	Are thorough tests provided to validate functionality?
<b>Problem Solving:</b>	Handling of potential challenges like dynamic content or failures?

### Instructions

Provide your work to [marc@deck.co](mailto:marc@deck.co) & [keith@deck.co](mailto: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

python

```
.
├── README.md
├── requirements.txt
├── src/
│   ├── __init__.py
│   ├── config.py
│   ├── scraper.py
│   └── tests/
│       ├── __init__.py
│       └── test_scraper.py
└── .gitignore
```

## Step-by-Step Instructions

### 1. **\*Setup the Project:\***

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

### 2. **\*Understand the Target Site:\***

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

### 3. **\*Basic Authentication:\***

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

### 4. **\*Data Extraction:\***

- 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
- Implement error handling for common issues like elements not found or network errors.
- 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 `pytest tests/` to ensure all functionality works as expected.

**\*\* END OF ASSIGNMENT \*\***