# COUPANG WEB AUTOMATION ASSIGNMENT REPORT

Name: Nelton   
Email: [neltontan@outlook.com](mailto:neltontan@outlook.com)  
Date: 20 Jul 2025

## 1. Overview

**Objective:** To build a robust web scraper that can extract product data from any e-commerce website, supporting dynamic content, pagination, and anti-bot protections.

**Sites Explored:**

* SHEIN (<https://sg.shein.com/pdsearch/Shoes%20For%20Men/>)
* Decathlon (<https://www.decathlon.sg/c/men/shoes.html>)
* Zalora (<https://www.zalora.sg/c/men/shoes/c-27>)

**Final Target Site:** Decathlon

**Reason for Selection:**  
Both SHEIN and Zalora implemented strong anti-bot mechanisms, including immediate CAPTCHA challenges and aggressive rate limiting. Decathlon, on the other hand, used dynamic content loading but did not aggressively block scraping attempts, making it the most feasible for this assignment.

## 2. Approach & Tools

**Programming Language:** Python 3.11

**Libraries & Tools:**

* Playwright (for headless browser automation and dynamic content handling)
* BeautifulSoup (for HTML parsing)
* CSV (for exporting structured data)

**Scraper Features:**

* JavaScript-rendered content support
* Pagination via "View More" button clicks
* Image, title, price, and product URL extraction
* CAPTCHA detection and graceful handling
* Retry logic and basic anti-bot delay simulation

## 3. Site Attempts & Challenges

| Site | Key Issues Encountered | Result | Time Spent |
| --- | --- | --- | --- |
|  |  |  |  |
| Zalora | CAPTCHA after 1-2 pages; content inaccessible via automation | Blocked | ~45 mins |
| SHEIN | CAPTCHA triggered immediately, blocking access | Blocked | ~15 mins |
| Decathlon | JavaScript-heavy but navigable with Playwright and delays | Successful | ~20 mins |

## 4. Iterative Code Versions

* **Version 1:** Static scraping with requests and BeautifulSoup (failed on JS content).
* **Version 2:** Selenium-based script (Zalora blocked after 1-2 pages).
* **Version 3:** Playwright-based script with dynamic content loading.
* **Version 4:** Improved Playwright script that clicks "View More" up to 10 times and extracts structured data into CSV.

## 5. Output Sample for Decathlon

Note: Zalora and Shein did not due to anti-bot protection

Fields extracted:

* Title
* Price
* Image URL
* Product URL

Final CSV output file: decathlon\_output.csv

Example record:

"Kalenji Run 100 Men's Running Shoes - Grey", "$14.90", "https://.../kalenji-run-100.jpg", "https://www.decathlon.sg/p/xyz"

## 6. Anti-Bot Mitigation Strategies

* **Delay Simulation:** Added randomized timeouts (e.g., 3 seconds) after each interaction.
* **Session Reuse:** Leveraged persistent Playwright context for scraping continuity.
* **CAPTCHA Detection:** Script skips page and logs warning if CAPTCHA container is detected.
* **Retry Logic:** Catches timeouts or loading errors, then gracefully continues.

## 7. Limitations & Future Enhancements

* **CAPTCHA Bypass:** Could integrate external CAPTCHA-solving services like 2Captcha.
* **Proxy Rotation:** Adding proxy support would help bypass IP-based rate limits.
* **Product Detail Pages:** Currently only list page items are scraped; future enhancements could follow product links and extract specifications or ratings.

## 8. Conclusion

This assignment demonstrated the ability to iterate through various web automation techniques, adapting to anti-bot challenges and dynamic content rendering. The final solution extracts more than 50 product listings from a JavaScript-heavy site while respecting scraping ethics and limits.

The tools and logic implemented are scalable and reusable for similar scraping tasks across other e-commerce domains.

Test Overview and Objective for Web Automation Engineer

Extract structured data from a given website, handle potential challenges such

as pagination or dynamic content, and present findings clearly. This test

evaluates practical skills in data extraction, anti-bot mitigation, and handling

complex web automation challenges that reflect real-world scenarios. Use

appropriate tools and communicate your approach and results effectively.

Estimated Time: 2-4 hours

Task Description

1. Core Web Automation Implementation

a. Basic data extraction - build a scraper to extract product

information from a website, handling at least 50 products while

implementing proper error handling.

b. Pagination Handling - the scraper must navigate through multiple

pages automatically, handling different pagination styles including

next buttons and page numbers.

2. Anti-bot Migration

a. Rate Limiting bypass - implement intelligent delays between

requests, exponential backoff for retries, user agent rotation, and

proper handling of 429 Too Many Requests responses.

b. Advanced Anti-bot handling - Implement proxy rotation, handling

JavaScript-rendered content, Captcha detection, and session

management.

3. Dynamic Content Handling

a. JavaScript-Heavy Sites - Use headless browser automation tools

like Selenium or Playwright to scrape JavaScript-heavy content,

properly waiting for dynamic content to load and handling AJAX

requests.

Deliverables

• Code/Scripts used for scraping and data extraction, with clear

instructions on how to run them

• README with set-up and usage instructions

• Extracted data file in JSON or CSV format

• Written report (1-2 pages) including your overall approach and design