Web Technologies

Homework 1 — Web Crawling and SLUview

Due on Sept 24th - 11:59pm *Fall 2025*

Introduction

The purpose of this assignment is to give you hands-on practice with the full lifecycle of web technologies:

- building a simple front-end site with HTML/CSS,
- crawling and scraping real-world data responsibly,
- processing scraped results into structured formats (CSV/JSON), and
- displaying those results inside your own mini web app.

You will start by building a small static website called **SLUview**, (same as in "review"), a lightweight Yelp-like competitior, and populate it with reviews you scrape from businesses from Yelp.com or TripAdvisor.com. This exercise will also train you to think about the ethics of web crawling.

Undergraduate students will complete the core tasks (front-end, scraping, parsing). Graduate students will additionally tackle a few more advanced programming tasks to simulate real-world data collection at scale.

1 For Graduate and Undergraduate Students

Q1. Build a Mini-Yelp Replica brand it SLUview using HTML/CSS.

Your task is to design and implement a simple static website called **SLUview** (as in "SLU review"). The purpose is to practice responsive layout and integration of scraped data into a basic front-end.

SLUview website Requirements:

(a) Create a responsive **3-column layout** using what we have learned in class, e.g., media queries:

- Column A: filters/navigation (e.g., category, price, rating). These do not need to be functional, but they must be styled and clearly labeled.
- Column B: featured businesses (at least 3 businesses, showing name, category, location, price, rating, and a short blurb).
- Column C: latest reviews (at least 5 reviews), populated from your scraping data.

(b) Include a stylesheet (called `slu.css`) with your own design choices and:

- A consistent color palette and readable typography.
- A responsive design: 3 columns on desktop, 2 on tablets, 1 on phones.

(c) Integrate scraped data using a file `data.json` or `reviews.csv` (generated from later questions):

- All students should use this file to pre-fill at least 5 reviews into Column C (copy the content manually into HTML "cards", or convert offline).
- **Graduate Students Only:** Write a short script that dynamically loads the data file (e.g., using JavaScript `fetch()`) and creates the review cards at runtime. We haven't covered much JavaScript in class but you are expected to do your own readings to figure out how to use it.

(d) Be creative! Your layout and style should not match the example images provided on canvas; it needs to be responsive, and clearly show the three sections with the required content. You must create your own CSS for this, not use a template.

What to submit for this question:

- A report that contains links to `index.html`, `styles.css`, and your `data.json/reviews.csv`. You should use hopper.slu.edu, github pages, or any other link that has a server running so that the grader can see the website live, not just the code.
- Screenshots of your site in desktop, tablet, and mobile views attached to the report.
- A short paragraph in your PDF explaining how you populated Column C (static vs. optional dynamic).

Q2. Understand Web Crawl Measurements.

Study in depth the following paper: "Towards Realistic and Reproducible Web Crawl Measurements" (`https://dl.acm.org/doi/10.1145/3442381.3450050`). If you cannot access ACM from off-campus, use the open PDF:
`https://brave.com/static-assets/files/realistic-www-2021.pdf`

Answer the following questions in your PDF report titled `First-Last-HW1.pdf`:

(a) What is this paper about? What are the research questions addressed in your opinion? Why there was the need to write this paper? Who do you think should care?

(b) What was the impact, if any, of running crawlers from the Cloud versus from residential hosts?

(c) How is the *bias score* defined in the paper? Why do you think there is a bias in crawling?

(d) What is the impact of browser automation frameworks on the accuracy of crawling results?

Hint: you should understand deeply this paper as questions similar to these will be present in one of the exams.

Q3. Web Scraping with Browser Plugin (Tutorial).

In this part you will practice scraping a real website using a point-and-click browser extension instead of writing code. The goal is to become familiar with how structured data can be extracted from web pages and then exported to files for later use.

Step 1. Install the plugin.

(a) Open Chrome (or Microsoft Edge).

(b) Go to `https://webscraper.io/` and install the official "Web Scraper" extension.

(c) Pin the extension so you can see it in your browser toolbar.

Step 2. Pick a unique listing.

(a) Navigate to `https://www.yelp.com` or `https://www.tripadvisor.com`.

(b) Choose a single business page (e.g., a restaurant, café, hotel, or attraction).

(c) Example of a valid page: `https://www.yelp.com/biz/blueprint-coffee-st-louis`.

(d) Each student must pick at random a different business in a different city in USA. The probability of having two students with the same business will be suspicious. Please write immediately on Slack the business you are picking so that nobody will pick the same. The instructor will also know when you have started to complete your homework once you have done that.

Step 3. Create a sitemap in Web Scraper.

(a) Open the Web Scraper extension and create a new sitemap. Use the URL of your chosen business page as the start URL.

(b) Define **selectors** (click on the review elements and tell the scraper what to extract).

(c) At a minimum, capture:

- Reviewer name or handle.
- Star rating.
- Review text.
- Date of the review.

(d) Optionally, also capture fields such as: review count, photos, "helpful" votes, or the business's overall rating.

Step 4. Run your scrape.

(a) Run the scraper directly from the extension.

(b) Make sure you only scrape your chosen page (not the entire domain).

(c) Let the scraper finish and review the data preview inside the plugin.

Step 5. Export your data.

(a) Export the results to CSV or JSON format (the plugin provides download options).

(b) Save the file as `reviews.csv` or `data.json`.

(c) Verify that you have at least 5 reviews in your exported file.

Step 6. Document your process.

(a) Take a screenshot of your selector configuration.

(b) Take a screenshot of the exported data preview.

(c) Include both screenshots in your submission PDF.

What to submit for this question:

- A URL where the grader can download the exported CSV/JSON file with at least 5 reviews. Do not submit the file itself as only PDFs will be accepted.
- Screenshots of your Web Scraper configuration and exported data preview.
- A short note (2 to 4 sentences) in your PDF report describing which fields you extracted and why you chose them.

Remark: This data will be reused in the next part of the assignment (building the SLUview website), so make sure you keep your file organized and easy to parse.

Q4. Fetching with curl.

In this part you will learn how to fetch and inspect webpages from the command line using `curl`, a standard tool that we have seen in class for transferring data over HTTP. This exercise will help you understand how servers adapt responses based on request headers and cookies.

Step 1. Study the curl manual.

(a) Before running any commands, open the curl manual:

- Run `man curl` on Linux/Mac, or `curl --help` on Windows.
- Skim through options for setting a User-Agent (`-A`), adding custom headers (`-H`), handling cookies (`--cookie-jar`, `--cookie`), and writing output to a file (`-o`).

(b) In your PDF report, include a short (3–5 line) summary of what you learned from the manual about how these flags work.

Step 2. Basic fetch.

(a) Use curl to fetch the HTML of your chosen Yelp or TripAdvisor listing:

```
curl -L "https://www.yelp.com/biz/REPLACE-ME/reviews" -o
   listing.html
```

(b) Open `listing.html` in a browser or text editor and confirm you captured the full page.

Step 3. Experiment with headers.

(a) **User-Agent:** Repeat your fetch with a custom User-Agent string, e.g.:

```
curl -L -A "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)"
    \
 "https://www.yelp.com/biz/REPLACE-ME/reviews" -o listing_ua
    .html
```

Compare `listing.html` vs. listing_ua.html. Did the server return different markup or block the request?

(b) **Other headers:** Add an `Accept-Language` header to request different languages. Example:

```
curl -L -H "Accept-Language: fr-FR" \
 "https://www.yelp.com/biz/REPLACE-ME/reviews" -o listing_fr
    .html
```

Open the file and note if menus or dates are localized.

Step 4. Experiment with cookies.

(a) Save cookies to a file when making a request:

```
curl -L --cookie-jar cookies.txt \
 "https://www.yelp.com/biz/REPLACE-ME/reviews" -o
    listing_cookies.html
```

(b) Reuse those cookies in a follow-up request:

```
curl -L --cookie cookies.txt \
 "https://www.yelp.com/biz/REPLACE-ME/reviews" -o
    listing_reuse.html
```

(c) Compare responses with and without cookies. Did banners (e.g., consent pop-ups) disappear? Did the server remember your preferences?

Step 5. Write a report with your reflection.

(a) In your report, answer:

- How did the output differ when varying User-Agent, Accept-Language, and cookies?
- Why do websites adapt content based on these headers? Did they actually adapt in your case? If not, why not? Investigate and explain.
- What are the ethical limits of using curl this way (e.g., impersonation, bypassing controls)? Discuss and be ready to rewrite this question in one of the exams.

(b) Limit your answer to about half a page (font-size 10 to 12pt).

What to submit for this question:

- At least 3 saved HTML files from your different curl runs (`listing.html`, `listing_ua.html`, `listing_fr.html` (for example), etc.).
- `cookies.txt` (if used).
- A short written summary (3–5 sentences per experiment) in your PDF report explaining what you observed.

Reminder: Use curl responsibly. Only fetch the single business page you selected. Do not attempt to crawl multiple pages automatically or bypass access controls.

For Graduate Students ONLY (undergraduate students can answer this question for 3 extra credit points): If you had to bypass access control, how would you do it? Do a little research and explain how you can prevent users from crawling multiple pages of your site creating a sort of denial of service attack. How can online businesses protect against unwanted crawling in your opinion?

Q5. Parsing with BeautifulSoup (or Equivalent Library).

In this part you will process the raw HTML you fetched with `curl` (saved as `listing.html`). The goal is to write a program that extracts structured data from the unstructured HTML of your Yelp/TripAdvisor page.

What tool to use?

- The recommended option is to use **Python** with the `BeautifulSoup` library, which is widely used for parsing HTML and XML.
- To install: `pip install beautifulsoup4 lxml`
- If you prefer another language, you may use an equivalent library:
  - **JavaScript/Node.js:** `cheerio` (jQuery-like API for HTML parsing).
  - **Java:** `jsoup` (popular for DOM parsing and web scraping).
  - **Go:** `goquery` (similar API to jQuery).
  - **Ruby:** `Nokogiri` (HTML/XML parser).
- Whatever language you choose, the assignment requirements remain the same.

Step 1. Inspect the page structure.

(a) Open `listing.html` in a browser.

(b) Use "Inspect Element" to find how reviews are structured (e.g., div classes for text, reviewer, rating).

(c) Write down the CSS selectors you will need.

Step 2. Parse reviews.

(a) Write a script that loads the HTML file.

(b) Select at least 5 review elements.

(c) Extract at least **6 different fields** (see list below).

(d) Store extracted fields in memory (arrays, objects, etc.).

Step 3. Fields to Parse (choose ≥ 6).

- Business info i.e., name, category, city/region, price range, overall rating, total review count.
- Review info i.e., reviewer handle, star rating, date, review text.

Step 4. Export results.

(a) Save your output to a structured file (`parsed.json` or `parsed.csv`).

(b) Each row/object should correspond to one review with all chosen fields.

Step 5. Reflection.

(a) In your PDF report, explain:

- Which selectors you used and why.
- Any difficulties (e.g., dynamic content, hidden fields).
- How you verified correctness (e.g., comparing with browser view).

What to submit for this question:

- A URL to your parsing scripts (`parse.py`, `parse.js`, etc. in a zip file).
- The exported CSV/JSON file with at least 5 reviews and ≥ 6 fields.
- A short reflection (5 - 6 sentences) in your PDF report.

2 For Graduate Students Only

Graduate students must complete the following questions:

G1. Mini Research on Responsible Use of `curl`

Conduct a short scientific-style investigation into responsible and ethical use of `curl` and similar HTTP tools. Your write-up should be 300–400 words and must cite at least two external sources (academic papers, technical blogs, or official documentation). Discuss what you like. Cite your sources.

G2. Advanced Programming Task: Multi-Page Scraper. Extend your Beautiful-Soup script into a small crawler that:

(a) Automatically follows pagination links to collect at least **3 pages of reviews** for your chosen listing.

(b) Stores results into a structured JSON or CSV file.

(c) Handles edge cases gracefully (e.g., missing fields, fewer reviews than expected, no next page).

(d) Includes a command-line argument to specify how many pages to scrape; the default value should be 3.

Your output should contain at least 15 reviews and be usable as the `data.json` file for SLUview.

Remark: Each student must work on a different pages. Do not share selectors or code verbatim. Instructors or TAs may request a live demo.

What to Submit

- SLUview site: `index.html`, `styles.css`, `data.json/reviews.csv`, screenshots.
- Paper deliverables: answers, slides, video link.
- Scraping artifacts: exported CSV/JSON, curl commands, saved HTML, Python scripts, parsed output.
- Reflection: comparison paragraph + ethics checklist.
- Graduate students: essay + advanced scraper with multi-page output.

Grading Criteria (100 points total)

Your grade will be calculated out of 100 points. Undergraduate students should complete questions Q1–Q5. Graduate students should complete Q1–Q5, G1, and G2, with adjusted weights so their total also equals 100 points.

Grading Criteria for Undergraduate Students (100 points total)

- **Q1. SLUview Website — 35 pts**
  Responsive 3-column layout, coherent CSS styling, and integration of at least 5 scraped reviews (static insertion acceptable).
- **Q2. Paper Study — 10 pts**
  Short-answer responses are correct, concise, and demonstrate understanding of the paper.
- **Q3. Web Scraper Plugin — 15 pts**
  Successful scrape with at least 5 reviews, clear screenshots of setup, and brief explanation of chosen fields.
- **Q4. `curl` Experiments — 15 pts**
  At least 3 runs with varied headers/cookies, correct use of options, and short reflection on differences and rationale.
- **Q5. Parsing with BeautifulSoup (or equivalent) — 25 pts**
  Script extracts at least 6 fields from at least 5 reviews, exports to CSV/JSON, and includes a brief reflection on selectors/validation.

Grading Criteria for Graduate Students (100 points total)

- **Q1. SLUview Website — 30 pts**
  Responsive 3-column layout, coherent CSS styling, and reviews dynamically loaded from `data.json` using JavaScript.
- **Q2. Paper Study — 5 pts**
  Short-answer responses are correct, concise, and demonstrate understanding of the paper.
- **Q3. Web Scraper Plugin — 10 pts**
  Successful scrape with at least 5 reviews, clear screenshots of setup, and brief explanation of chosen fields.
- **Q4. `curl` Experiments — 10 pts**
  At least 3 runs with varied headers/cookies, correct use of options, and short reflection on differences and rationale.
- **Q5. Parsing with BeautifulSoup (or equivalent) — 15 pts**
  Script extracts at least 6 fields from at least 5 reviews, exports to CSV/JSON, and includes a brief reflection on selectors/validation.
- **G1. Mini Research on `curl` Ethics — 15 pts**
  300–400 word report with at least two cited sources; covers impersonation, cookies, server load, privacy, and legal/ethical limits.
- **G2. Advanced Multi-Page Scraper — 15 pts**
  Script handles pagination for at least 3 pages, collects at least 15 reviews, and stores results in CSV/JSON with basic error handling.