Web scraping + Mini-UI Exercise (3–4 hours)

Goal: build a small scraper that collects product data from an intentionally scrape-friendly site and a lightweight frontend to explore it.

Due Date: within 24 hours of email sent.

Allowed Targets (choose one):

- Books to Scrape (https://books.toscrape.com)
- Quotes to Scrape (https://quotes.toscrape.com)
  (these are purpose-built for practice scraping; please do **not** scrape other sites.)

AI Tools:
You may use ai tools to assist you in making this project; however, you will be expected to be able to explain the entire project, there will be a live coding portion of the interview to extend the project, and we would like you to be upfront on which tools you used and why.

Deliverables

1. **Scraper** (Node or Python)
    - inputs: start URL(s); max pages to crawl; polite delay (ms).
    - fetch pages, follow pagination, extract core fields.
        - **Books**: title, price, availability, rating, category, product page URL.
        - **Quotes**: quote text, author, tags, quote page URL; (bonus: author details).
    - write results to `data/items.jsonl` (1 JSON object per line).
    - handle:

        - robots.txt check (log what you enforce),
        - rate limiting (configurable delay, e.g., 500–1500ms),
        - basic retry with exponential backoff,
        - graceful stop on max pages,
        - dedup by URL or stable key.
    - a `--dry-run` flag that logs what *would* be crawled without saving.
2. **frontend** (React preferred; Vue and Angular ok)
    - load `items.jsonl` (or a served JSON) and show a table/grid.
    - features:
        - text search (title/quote),
        - filter by category/tag,
        - sort by price/rating (Books) or by author/tag count (Quotes),
        - client pagination.
    - include a tiny visualization:
        - Books: bar chart of count by category or rating.
        - Quotes: bar chart of count by tag or top authors.
    - show row click → detail panel (fields you scraped).
3. **tests + docs**
    - scraper unit tests for “parse page → structured item” (use stored HTML fixtures).
    - simple integration test for pagination logic (mock or a limited crawl).
    - README with:
        - how to run scraper & UI,

        - your design decisions,
        - what you'd do with more time,
        - known limitations.

constraints & expectations

- honor robots.txt intent, add a custom `User-Agent`
- **tools:** pick either
    - **Node**: `node-fetch/undici`, `cheerio`, optional `puppeteer` (only if needed),
    - **Python**: `requests/httpx`, `beautifulsoup4`, optional `playwright` (only if needed).
- prefer simple HTML parsing first; use a headless browser *only if necessary* and explain why.
- keep it under 4 hours; it's okay to skip "nice-to-haves" but document what you'd add.

submission

- GitHub repo link (or zip) with commit history.
- include a short loom/video (≤5 min) demoing:
    - run scraper (small crawl),
    - start UI, show filters/search/sort,
    - walk through code briefly.

starter structure (suggested)

```
webscrape-intern/
  scraper/
    package.json or pyproject.toml
    src/
      main.(ts|js|py)
      fetcher.(ts|js|py)
```

```
    parser.(ts|js|py)
    pagination.(ts|js|py)
    robots.(ts|js|py)
    types.(ts|js|py)
  tests/
    fixtures/
    test_parser.(ts|js|py)
  data/           # generated
  README.md
ui/
  package.json
  src/
    App.tsx
    components/
      Table.tsx
      Filters.tsx
      Chart.tsx
    lib/loadData.ts
  README.md
README.md         # top-level: how to run both
```

helpful commands (example)

- Node scraper:
    - `npm run crawl -- --site=books --start=https://books.toscrape.com --maxPages=5 --delayMs=700`
    - `npm run crawl -- --dry-run ...`
- Python scraper:
    - `python -m scraper.main --site=quotes --start=https://quotes.toscrape.com --max-pages=5 --delay-ms=700 --dry-run`
- UI:
    - `npm run dev` (Vite) → visit http://localhost:5173

bonus (optional)

- resume crawl (append without duplicates),
- small CLI `--concurrency=N` with queue (be mindful of site load),
- “schema version” field in output, and a simple migration note.