voiceflip

Technical Challenge – Web Scraping & Automation Analyst

Objective: Evaluate your ability to design and implement a dynamic scraping system that downloads files, detects changes, and runs locally in an automated fashion.

---

1. Challenge Description

- **Scraping a website:** Choose a real public site with pagination or infinite scroll (e.g., Mercado Libre, AliExpress, Plusvalia, Inmuebles24). This will be used exclusively for structured data extraction.
- **File Management:** Use the static HTML site provided in the attached ZIP file to simulate downloadable files (PDFs, images, etc.). Mount it locally (e.g., using `python -m http.server`) and manually edit, replace, or delete files to test your detection logic.
- **Extract structured data:** Gather fields like title, price, date, etc.
- **Download files:** Store them in a local folder and track file hashes.
- **Persistence:** Save data in PostgreSQL and manage associated file versions.
- **Change detection:**
    - New record → insert + alert
    - Modified record → update + alert
    - Deleted record → remove from DB and delete files
    - File content changed (hash differs) → replace file
    - File removed from source → delete locally
- **Automation:** Run the scraper every hour using one of the following:
    - `apscheduler`
    - cron
    - Simulated Azure Function using `func start`
- **Resilience:** Include structured logging and proper error handling to avoid runtime failure.
- **(Bonus)** Use a language model (e.g., OpenAI) to generate or adapt CSS/XPath selectors dynamically.

2. Technical Requirements

Area	Minimum Requirements
Language / Framework	Python 3.9+, Scrapy + Playwright (or justify if using Selenium/Requests-HTML)
Persistence	PostgreSQL
File Handling	Download, hash check (SHA-256), replace/delete logic
Automation	APScheduler, cron, or `func start` for local function simulation
Logging / Errors	JSON-structured logs, resilient exception management
Testing	Script to demonstrate a short run that detects data and file changes
Documentation	README with setup steps, environment variables, and architecture diagram

---

3. Deliverables

1. **GitHub Repository**:
   - Source code, `requirements.txt` or `pyproject.toml`
   - Scheduler script or `func start` setup instructions
   - `docs/` folder with quick-start guide
2. **Short demo (video)**:
   - First run: scrape, populate DB, download files
   - Second run: detect record and file changes, generate alerts
3. **README**:
   - Local setup, environment variables, and design explanation
4. **(Optional Bonus)**:
   - Notebook or script using LLM to generate/adapt selectors

Authentication using API Key

Target URI:
https://voiceflip-openai.openai.azure.com/openai/deployments/gpt-4o-mini/chat/completions?api-version=2025-01-01-preview
API_KEY:
FwsUIhIZedFYxW7nGYwKgoJsMXYAH62OE4QThqLrwtKuCc5m17AjJQQJ99BEACYeBjFXJ3w3AAABACOGfF7h

```
import os
from openai import AzureOpenAI

client = AzureOpenAI(
    api_version="2024-12-01-preview",
    endpoint="https://voiceflip-openai.openai.azure.com/",
    credential=AzureKeyCredential("<API_KEY>")
)
```

Basic Example

```
import os
from openai import AzureOpenAI

endpoint = "https://voiceflip-openai.openai.azure.com/"
model_name = "gpt-4o-mini"
deployment = "gpt-4o-mini"

subscription_key = "<your-api-key>"
api_version = "2024-12-01-preview"

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    api_key=subscription_key,
)

response = client.chat.completions.create(
    messages=[
        {
            "role": "system",
            "content": "You are a helpful assistant.",
        },
        {
            "role": "user",
            "content": "I am going to Paris, what should I see?",
        }
    ],
    max_tokens=4096,
    temperature=1.0,
    top_p=1.0,
    model=deployment
)

print(response.choices[0].message.content)
```