AI Automation Engineer Exercise

You have up to 2 hours to complete this exercise. Focus on functionality over perfection.

Assistance from AI is acceptable but vibe coding is not.

🏁 When you are done or time is up, please push your code to a public GitHub repo and send us the link to the root directory.

Part 1: Core Selenium Automation (45-60 minutes)

- Use Python and Selenium to automate scraping the first 5 product listings from https://www.amazon.com/s?k=laptops (or a similar public site if Amazon blocks)
    - Fallback to https://fakestoreapi.com/products via requests if Selenium setup is tricky.
    - Attempt Selenium first. Only use the fallback if you can't access Amazon after two attempts.
- For each product, extract: Title, Price, Rating (if available), and URL.
- Store the data in a list of dictionaries and print it as JSON.
- Handle common issues: Use explicit waits (e.g., WebDriverWait), manage headless mode, and add basic error handling (e.g. for no elements found).
- Requirements:
    - Use Chrome WebDriver (assuming it's installed).
    - No hard-coded delays; use proper waits.
    - Keep it under 50 lines of code if possible (not including comments).

Part 2: AI Enhancement (30-45 minutes)

- Integrate an AI API to enhance the scraped data.
- Example enhancements (choose one or more):
    - Use AI to categorize products (e.g., "budget", "gaming", "professional") based on titles/descriptions.
    - Summarize ratings/reviews if available (e.g., generate a one-sentence sentiment analysis).
    - Make the script more robust: Use AI to generate dynamic XPath/CSS selectors if the page changes (simulate by providing a "broken" locator).
    - Creatively come up with your own enhancement
- Output the enhanced data (e.g., original data + AI-generated fields) as JSON.
- If using OpenAI, assume an API key is provided (or use your own).
    - Keep prompts simple and efficient.

Bonus (if time allows): Add a command-line argument to search for different keywords (e.g., python script.py --query="headphones").

Evaluation Criteria:

- Code runs without errors.
- Clean, readable code with comments.
- Effective use of Selenium best practices.
- Creative/relevant AI integration that adds value.
- Edge case handling (e.g. no results found).

Reminder: When you are done or time is up, please push your code to a public GitHub repo and send us the link to the root directory.