Coding Challenge: Mercari Japan AI Shopper

Challenge Overview

Your goal is to develop a Python-based AI agent that can:

1. Understand User Requests: Accurately interpret natural language requests from users specifying what they want to buy on Mercari Japan.
2. Effective Mercari Search: Perform effective searches on Mercari Japan using appropriate keywords and filters derived from the user's request.
3. Data Retrieval: Efficiently extract relevant product information from Mercari Japan search results, including product names, prices, conditions, and seller ratings (optional).
4. Reasoned Recommendations: Analyze the retrieved data and select the top 3 product options that best meet the user's needs, providing clear and concise reasons for each recommendation.
5. User-Friendly Output: Present the recommendations to the user in a well-structured, easy-to-understand format

Technical Requirements

- Programming Language: Python 3.10 or later
- LLM API: You may use either the OpenAI API or the Anthropic Claude API
    - Recommended: Claude 3.5 Sonnet is recommended for its balance of performance and cost-effectiveness.
- Tool Calling Implementation: Your solution must implement an agent using the tool-calling mechanism. You can refer to the documentation below, but you are not limited to them.
    - OpenAI's function calling API
    - Anthropic's tool use API
- Web Scraping: You are permitted to use libraries like `requests` and `Beautiful Soup` or `Scrapy` for web scraping, if necessary. Alternatively, you can use browser automation tools like `Selenium` or `Playwright` if it offers a more robust solution for data extraction.

- External Libraries: You are free to use any other relevant Python libraries that you find helpful. Clearly document the purpose of each library in your code.
- Framework Restrictions:
    - No Third-Party AI Agent Frameworks: To ensure a focus on fundamental agent development skills, you are not permitted to use high-level AI agent frameworks. This includes (but is not limited to) LangChain, LangGraph, LlamaIndex, and similar libraries designed to simplify agent creation. However, you are welcome to study these frameworks or implementations to help you get onboarded with agent development.
    - Permitted Libraries: You may use libraries for web scraping, data handling, and the official SDKs provided by OpenAI (e.g., openai) or Anthropic (e.g., anthropic-python) for direct interaction with their LLM APIs.

Deliverables

Please submit your solution as a ZIP file containing:

1. README.md: A Markdown file that includes:
    - Overview: A brief description of your agent's architecture and approach.
    - Setup Instructions: Detailed steps on how to install dependencies
    - Usage Instructions: Clear examples of how to run the agent and interact with it.
    - Design Choices: Explanation of any significant design decisions you made, especially regarding the use of the LLM and web scraping techniques.
    - Potential Improvements: Any ideas you have for further enhancing the agent's capabilities.
2. Source Code: Well-commented Python code for your agent.
3. requirements.txt: A file listing all the required Python packages for your agent.

Good luck, and we look forward to seeing your innovative solutions!