Product Requirements Document (PRD) for Web Scraper

Title: Multi-Website Product Price Comparison Scraper

Objective:

Develop a Python-based web scraper using BeautifulSoup that extracts product prices from e-commerce websites (e.g., Amazon, Myntra, Flipkart etc) to facilitate price comparison for users.

Background:

Online shoppers often visit multiple e-commerce websites to compare the price of a specific product before making a purchase. A tool that can automatically scrape and present prices from multiple websites would save time and potentially lead to cost savings.

Functional Requirements:

Input Interface:

Users should be able to enter the URL of the product (or the name of the Product) they wish to compare.

Websites to be Scrapped:

- Amazon
- Myntra/Flipkart (Note: Additional websites can be added in future versions)

Data Extraction:

Extract the following details:

- Product Name
- Product Price
- Website Source

Output:

Display the extracted data in a structured format to facilitate easy comparison. This can be a table that lists out prices from different websites.

Error Handling:

In case a product is not found on a website, provide a message indicating the same.
If the scraper encounters any issues (e.g., website structure changes), it should fail gracefully and notify the user.

Frequency:

The scraper should run on-demand, i.e., when a user provides input.

Non-Functional Requirements: [Optional]

Performance:

The scraper should provide results in a reasonable amount of time (ideally, within seconds).
Scalability:

The architecture should support adding more websites in the future without a major overhaul.
– Extensibility : We should be able to add more features to the product in future.

Usability:

The interface should be intuitive and easy to use.

Maintainability:

Code should be well-documented to allow for easy updates, especially if website structures change.

Constraints:

Rate Limiting: [Optional]

Some websites may impose rate limits or temporarily block IP addresses that send too many requests in a short time. Implement mechanisms to detect and respect these limits.

Technology Stack:

Language: Python
Libraries:

- BeautifulSoup4 for HTML parsing.
- Requests for making HTTP requests.

Milestones:

M1: [1.5-2 hours]

- Set up the project environment [Python, BS4, IDE, required libraries] and required libraries.
- Implement a basic scrapper for Amazon.

M2: [1-2 hours]

- Extend the scraper to support Myntra/Flipkart.
- Integrate an input interface for user product requests.

M3: [1 hour]

- Implement output display in a structured format.
- Add error handling mechanisms.
- Testing and QA.

Future Considerations:

- Implement a caching mechanism to avoid re-scraping products within short time intervals.
- Extend the scraper to support more e-commerce websites.
- Develop a GUI or web interface for a more user-friendly experience.