Crawlability and Robots.txt Analysis Report for Amazon

1. **Introduction**

This report documents the crawl permissions and restrictions for the Amazon website, based on an analysis of the robots.txt file and an examination of sitemap URLs, RSS feeds, and API endpoints. The goal is to ensure polite and compliant web crawling for data collection and analysis.

1. **Robots.txt Rules**

* The robots.txt file specifies which paths are allowed or disallowed for web crawlers (user-agents).
* The wildcard user-agent \* applies to all bots unless otherwise specified

**Disallowed Patterns (as identified by crawl.py**): https://www.amazon.com/gp/registry/wishlist/XYZ/reserve, https://www.amazon.com/review/common/du, https://www.amazon.com/exec/obidos/account-access-login

**Allowed Exceptions:** https://www.amazon.com/gp/wishlist/universal, https://www.amazon.com/gp/wishlist/ipad-install

1. **Sitemap URl:**

<https://www.amazon.com/sitemap.xml>

1. **Crawl Delay:**

* No explicit Crawl-delay directive is specified in the robots.txt
* It is recommended to implement a polite crawling delay, such as 2 seconds between requests, to avoid overloading the server

1. **RSS Feeds and API Endpoints**

**5.1 RSS Feeds**

* No information on official RSS feeds for Amazon was provided in the given context.

omepage & Category Pages: https://www.amazon.com/, https://www.amazon.com/s?k=smartphones, https://www.amazon.com/gp/bestsellers, https://www.amazon.com/gp/new-releases

Product Pages: https://www.amazon.com/dp/B09G3HRMVP, https://www.amazon.com/dp/B00005N5PF

Wishlist & Lists: https://www.amazon.com/wishlist/universal, https://www.amazon.com/wishlist, https://www.amazon.com/gp/wishlist

Reviews: https://www.amazon.com/product-reviews/B09G3HRMVP, https://www.amazon.com/gp/customer-reviews/write-a-review.html

Sign-in & Cart: https://www.amazon.com/gp/cart, https://www.amazon.com/ap/signin

Media & Video: https://www.amazon.com/gp/video/library, https://www.amazon.com/gp/video/search

Help & Contact: https://www.amazon.com/gp/help/customer/display.html, https://www.amazon.com/hz/contact-us

Deals & Offers: https://www.amazon.com/gp/goldbox, https://www.amazon.com/s?k=deals

**5.2 API Endpoints**

* API paths are typically disallowed in robots.txt and should not be accessed by crawlers
* Attempts to crawl or scrape private API endpoints can violate site policy and may result in blocking.
* It is generally recommended to use sitemaps and RSS feeds for collecting content data instead of private APIs

1. **Crawling Implementation**

**A-Main Content Crawling (using api.py)**

The api.py script utilizes Selenium to scrape content from Amazon, specifically targeting article links from a given URL.

* User-Agent: A fake\_useragent library is used to generate a random User-Agent string for requests. This helps in simulating human-like Browse behavior.
* Selenium Options: The script configures Selenium Chrome options to avoid detection as an automated bot, including disabling webdriver flag, disabling various browser features, and setting window size.
* Scrolling and "Load More": The script includes functions to scroll down the page and click "Load More" buttons to ensure dynamic content is loaded.
* Link Extraction: It extracts article links from the HTML content using BeautifulSoup, specifically looking for <a> tags within article elements.
* Retry Mechanism: The selenium\_scrape\_with\_retry function implements a retry mechanism for loading URLs, attempting up to 3 times with a 5-second delay between retries.
* Saving Data: Extracted links are saved to a CSV file (e.g., amazon.csv) along with the page title.

**B- Articles Crawling (using ContentExtractor.py)**

The ContentExtractor.py script focuses on extracting product data from Amazon for "home essentials" keywords.

* Keyword-based Search: It iterates through a list of home essentials keywords (e.g., "kitchen appliances", "blenders") and constructs search URLs.
* Pagination Support: The crawl method supports pagination by iterating through a specified number of pages (max\_pages) for each search query.
* Data Extraction: It extracts product information such as name, price, customer ratings, category, and image URL using BeautifulSoup.
* Error Handling: The script includes basic error handling to skip items that cause issues during extraction.
* Delay: A random delay between 2 and 5 seconds is introduced between page requests.
* Saving Data: All collected product data is saved to a CSV file (e.g., data/home\_essentials\_products.csv).

**C- JavaScript Content (using api.py)**

The api.py script demonstrates handling JavaScript-rendered content using Selenium. It ensures effective scraping of dynamic pages through techniques like:

* Dynamic Scrolling: Scrolling down the page to load more content.
* Clicking "Load More": Interacting with "Load More" buttons to reveal additional content.
* Stealth Techniques: Using get\_selenium\_options and remove\_webdriver\_flag functions to avoid detection as a bot.