\*\***Objective**: Build a web scraping system for Amazon's top-selling products in selected categories with configurable scraping frequency.

**\*\*Description:\*\***

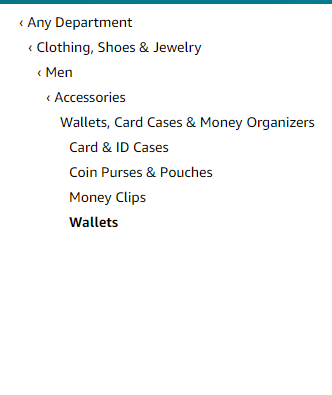
We need to develop a web scraping solution that can collect data from Amazon's product pages for top-selling items in specific categories. The system should allow us to configure the scraping frequency, enabling us to scrape the data twice a day or as per our requirements. Should bypass advanced website anti-boy systems.

**\*\*Technical Requirements:\*\***

**1. \*\*Web Scraping Framework:\*\*** Implement a web scraping framework or library (e.g., BeautifulSoup, **Scrapy**, Selenium) to extract data from Amazon product pages. Before selecting the web scraping framework to discuss options with client.

**2. \*\*Category Selection:\*\*** Create a mechanism to specify the target product categories. This can be done by providing category URLs or category keywords. Should have the ability to also exclude categories and add selected categories currently being used by Amazon.

The leaf category should be used. In the example below, “Wallets” is the leaf category - no further subcategories.



**3. \*\*Top-Selling Items:\*\*** Looking to track top selling product per category. We are not looking to track all items in category, but limited count of items, which should be a user configurable parameter (e.g., track top 50, 100, etc.)

[**https://www.amazon.com/Best-Sellers/zgbs**](https://www.amazon.com/Best-Sellers/zgbs)

**4. \*\*Type of Data to Extract:\*\*** Extract relevant information from product pages: ASIN, Best Sellers Rank (BSR), # of product sold in last month, product name/title, product brand, price, rating, number of customer reviews, country of origin, customer review (2-3 stars lower), product description, product specification, product image, product category .

**5. \*\*Configurable Parameters:\*\***

**- \*\*Scraping Frequency:\*\*** Develop a configuration option to set the scraping frequency (e.g., twice a day, once a day, etc.). Use a scheduler or cron job to automate this process. Should be able to run on multiple servers wherein each server works on a given category this way to split up the scraping load.

**- \*\*# of Products to Scrap Per Category :\*\*** Develop a configuration option that lets you select a number of products to scrape in a given category (e.g., scrape the wallets category for top 500 items, and a later date allowing for scrapping of top 200 items).

**- \*\*Product Category Selection :\*\*** Develop a configuration option that lets you select which product categories should be scrapped.

**- \*\*Error Handling:\*\*** Implement error handling and logging mechanisms to address issues like CAPTCHA challenges, page structure changes, or connection problems. Log fail crawls for later failure analysis.

**6. \*\*Proxy Support:\*\*** Incorporate proxy support to avoid IP bans and ensure reliable scraping.

**7. \*\*User-Agent Rotation:\*\*** Rotate user-agent headers to mimic different web browsers and reduce the chance of detection.

**8. \*\*Rate Limiting:\*\*** Implement rate limiting to prevent overloading Amazon's servers and minimize the risk of being blocked.

**9. \*\*Data Storage:\*\*** Store the scraped data in database format, where it could be aggregated, analyzed with other tools for analytics.

**10. \*\*Monitoring and Alerts:\*\*** Set up monitoring for the scraping process and configure alerts for any issues or failures.

**11. \*\*Testing and Maintenance:\*\*** Will be determined at a later phase, once system roadpath is determined.

**\*\*Documentation:\*\***

Provide documentation on how to use, configure, and maintain the scraping system.