

ext_aeid-5482-CartierAsk_v13-1

Analysis & Design Document

Date – 26 Oct 2022

Table of Contents

Topic	Page No.
1. Scope of work	3
2. Solution Approach.....	3-4
3. Script Development Flow.....	5
4. Technology Considerations.....	6
5. Base Collector Code.....	7-8
6. Template Parameters & Description.....	9
7. Risk & Dependencies.....	10

1. Scope of work

Scrap the below data from SITE: <https://www.cartier.com/en-us/home>

1. Product ID
2. Product Name
3. Product Size
4. Product Price
5. Product Availability
6. Product Details (Optional)

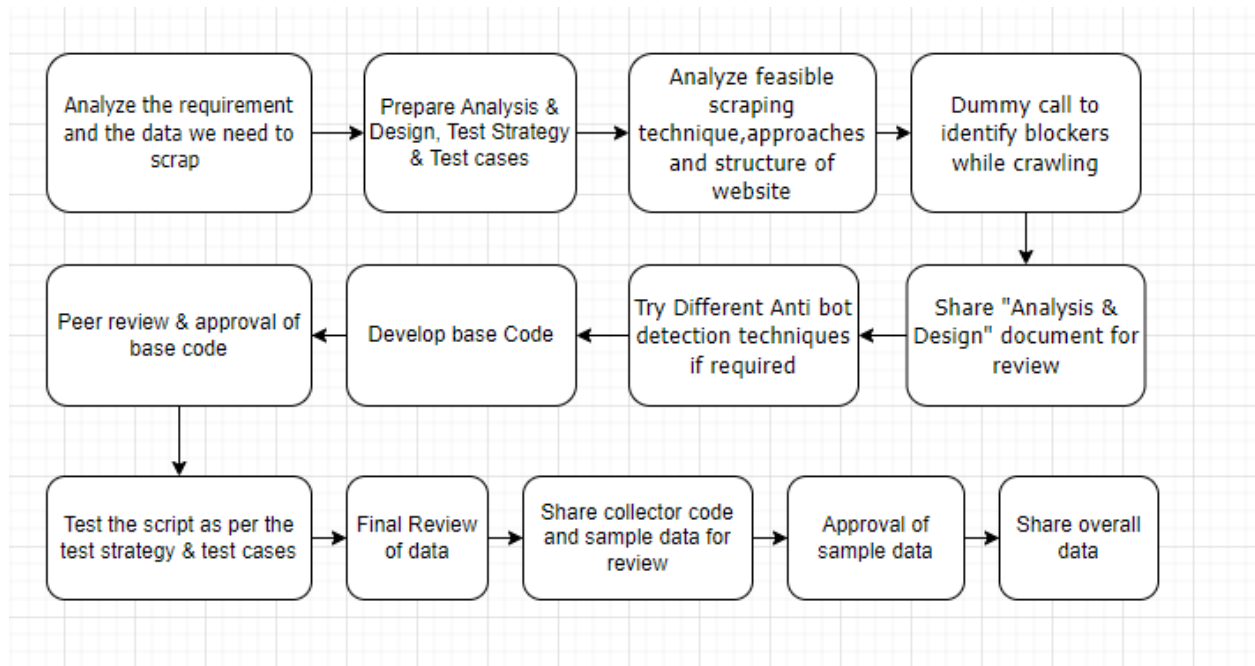
- Handbags data is priority where as other categories are good to have.
- Need data for other countries.

2. Solution Approach

We are following the below steps to develop the script as per the requirement

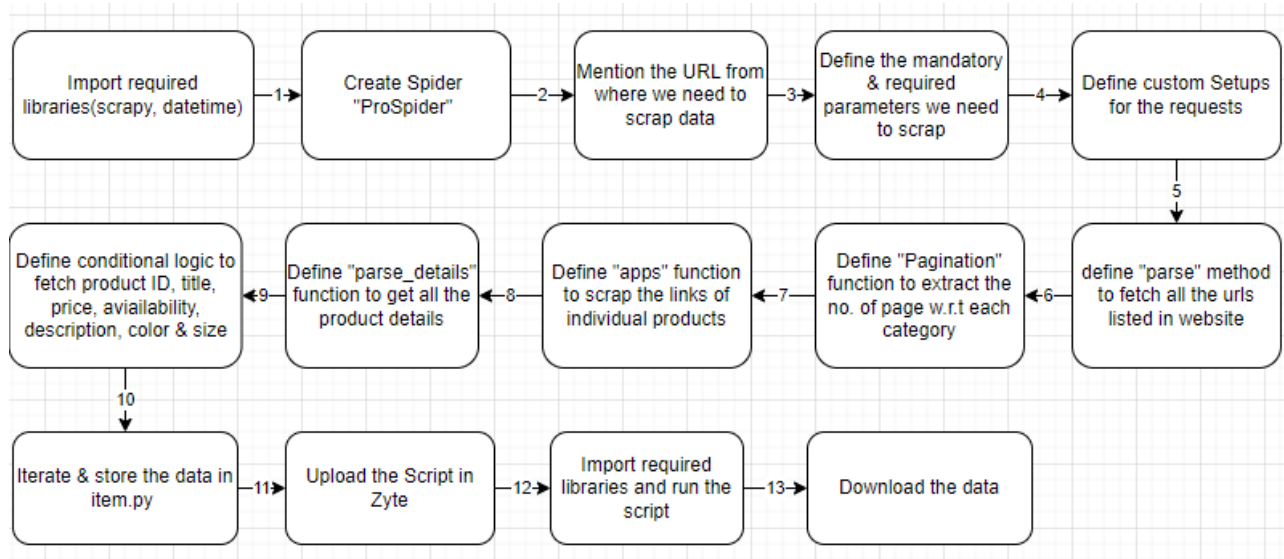
- The website is operational in 68 countries, hence the collector code is needed for all the countries.
- We are fetching the required details for each product.
- Checked the javascript data (the data we get from AJAX calls) with the help of view page source.
- We are fetching all category links then fetching the product links from where we are scraping the required product details.

Analysis & Design Document



3. Script Development Flow

Below steps are followed to create spider



4. Technology Considerations

Custom signup - Not required

Programming Language - Python

Framework - Scrapy

Tool - Zyte

Functions & Libraries used - datetime, scrapy-user-agents

Storage (Database) - Zyte Cloud

Deployment Requirements

- Install all the required libraries in Zyte Cloud

Logging considerations

- No logging is required
- No CAPCTHA authentication required

Proxy Details

- We are using user agent to avoid getting blocked, this is present in settings.py file.

5. Base Collector Code

File name - Cartier.py

Here we are scraping the data as per the requirements

Step 1 - Importing required libraries

```
from scrapy import Request
import scrapy
import json
from ..items import CartierItem
from datetime import datetime
```

Step 2 - Here we are defining the custom details

```
custom_settings = {
    'SCHEDULER_PRIORITY_QUEUE': 'scrapy.pqueues.DownloaderAwarePriorityQueue',
    'REACTOR_THREADPOOL_MAXSIZE': '20',
    'LOG_LEVEL': 'INFO',
    'RETRY_ENABLED': 'False',
    'DOWNLOAD_TIMEOUT': '1000',
    'REDIRECT_ENABLED': 'False',
    'AJAXCRAWL_ENABLED': 'True',
    'CONCURRENT_REQUESTS_PER_DOMAIN': '2',
    'DNS_RESOLVER': 'scrapy.resolver.CachingThreadedResolver',
    'DUPEFILTER_CLASS': "scrapy.dupefilters.BaseDupeFilter",
    'AUTOTHROTTLE_ENABLED': 'False'
}
```

Step 3 - Here a spider named "ProSpider" is created and start url of the website are defined that we are crawling

```
class ProSpider(scrapy.Spider):
    name = 'Cartier'
    headers = {
        'User-Agent': "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) "
        "Chrome/106.0.0.0 Safari/537.36"}

    start_urls = ['https://www.cartier.com/en-us/home']
```

Step 4 - Here we are defining the mandatory data

```
site = 'https://www.cartier.com'  
execution_id = '621291'  
feed_code = 'aaid5482'  
record_create_by = 'aaid5482_cartier'  
record_create_date = datetime.now()  
source_country = 'USA' this will be dynamic based on countries
```

Step 5 - Here we are fetching all links available on the website

```
def parse(self, response):
```

Step 6 - Here we getting the total no. of pages that category have

```
def pagination(self, response):
```

Step 7 - Here we are fetching the script and links of individual products

```
def apps(self, response):  
    data = response.xpath('//script[@type="application/ld+json"]/text()').get()
```

Step 8 - Here we are defining parse function. Inside this function we are writing code for crawling the details of each product

```
def parse_details(self, response):  
    items = CartierItem()  
    details = response.xpath('//script[@type="application/ld+json"]/text()').get()
```

Step 9 - yielding all items here

```
    yield item
```


6. Template Parameters & Description

The template contains the data that is scraped as per the ranking of newly listed products.

For the parameters where **mandatory** is mentioned, this is mandatory parameters as per the required template.

For the parameters where **Required** is mentioned, this is parameters needed as per the requirement document.

Below are the parameters that we are scraping and their description

1. **Context_identifier (Mandatory)** - We are capturing the hierarchy of product in a website
2. **Execution_id (Mandatory)** - Execution id will be taken automatically from zyte.
3. **Feed_code (Mandatory)** - This is hardcoded as project name.
4. **Availability (Required)** - This we are getting from website
5. **Available_Colors_OR_Size (Required)** - This we are getting from website
6. **Description (Required)** - This we are getting from website
7. **Price_Currency (Required)** - This we are getting from website
8. **Product_Id (Required)** - This we are getting from website.
9. **Product_Price (Required)** - This we are getting from website.
10. **Product_Title (Required)** - This we are getting from website.
11. **Record_create_by (Mandatory)** - This is hardcoded with spider name
12. **Record_create_dt (Mandatory)** - This is the timestamp for capturing the data.
13. **Site (Mandatory)**- This is hardcoded.
14. **Source (Mandatory)** - This is the link of the individual product.
15. **Source_country (Mandatory)** -This is hardcoded as per the specific country.
16. **Type (Mandatory)** - This is hardcoded.

7. Risks and Dependencies

Below are the identified risks and their possible solutions:

Risk	Mitigation
Risk of getting blacklisted/blocked/IP restrictions due to security/network policies on the web server.	we need to control the concurrency & use different proxy methods.
If the semantic code/markup of the website changes, the script will have a possibility of failure.	Identify the changes in the semantic code/markup of the website and modify the script accordingly.