ext_aeid5473-ChanelAsk_v13

Analysis & Design Document

Date – 29 Oct 2022

Table of Contents

Topic **Page No.**

1. Scope of work

Scrap the below data from SITE: https://www.chanel.com/ for all the operational countries.

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

2. Solution Approach

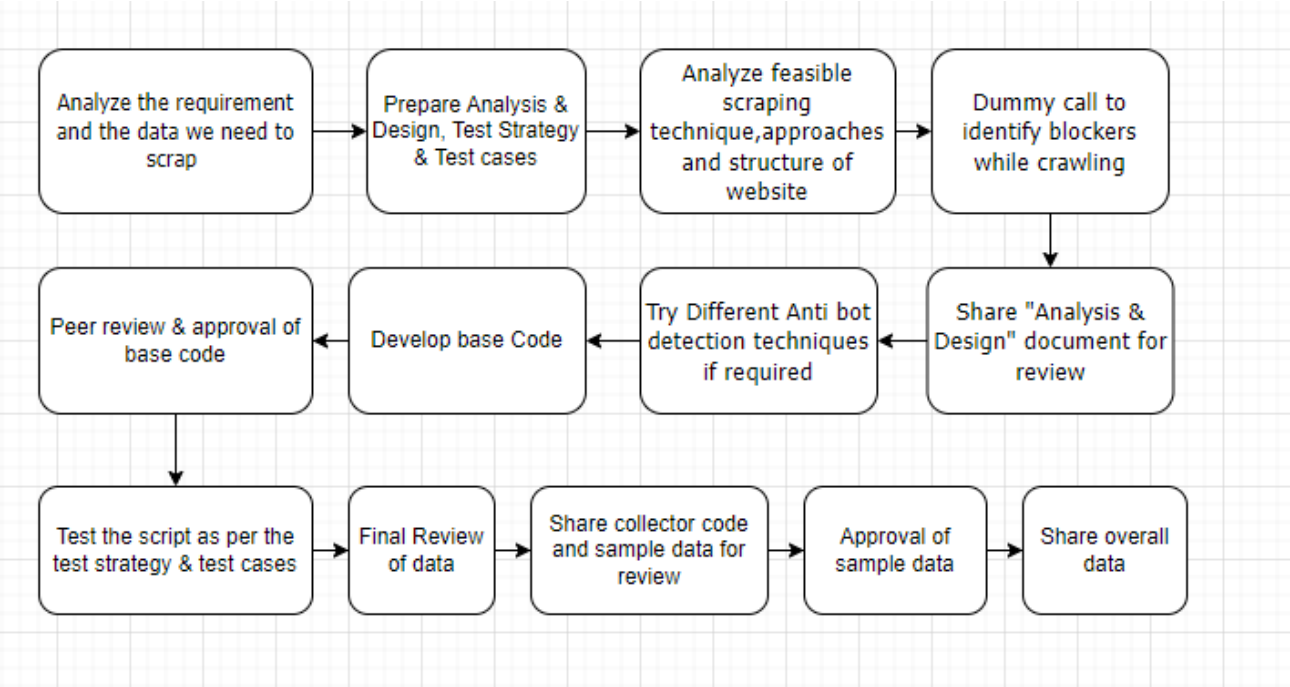
We are following the below steps to develop the script as per the requirement

- As the website is operational in many countries, we have created the collector code for 47 countries listed below.
- We checked which categories have the products that we need to scrap and then locate the URLs of that category.
- 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.

Country	Link	Spider
Albania , EN	https://www.chanel.com/al/	Done
Australia , EN	https://www.chanel.com/au/	Done
Austria , EN	https://www.chanel.com/at/	Done
Belgium , EN	https://www.chanel.com/be-fr/	Done
Bosnia and Herzegovina ,EN	https://www.chanel.com/ba/	Done
Brazil ,	https://www.chanel.com/br/	Done
Bulgaria, EN	https://www.chanel.com/bg/	Done
Canada , EN	URL is not operational	
Croatia , EN	https://www.chanel.com/hr/	Done
Czech Republic ,EN	https://www.chanel.com/cz/	Done
Denmark , EN	https://www.chanel.com/dk/	Done

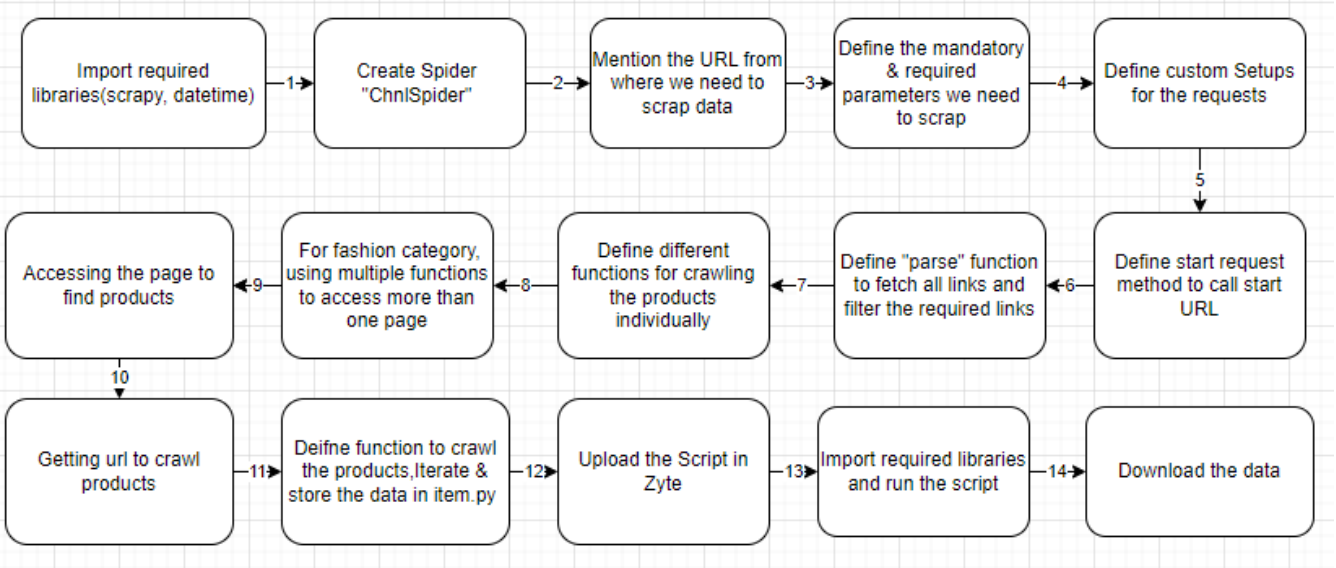
Estonia , EN	https://www.chanel.com/ee/	Done
Finland , EN	https://www.chanel.com/fi/	Done
France, EN	https://www.chanel.com/fr/	Done
Germany, EN	https://www.chanel.com/de/	Done
Greece, EN	https://www.chanel.com/gr/	Done
Hong Kong S.A.R ,EN	https://www.chanel.com/hk-en	Done
Hungary, EN	https://www.chanel.com/hu/	Done
Italy,EN	https://www.chanel.com/it/	Done
Japan,	https://www.chanel.com/jp/	Done
Kingdom Of Saudi Arabia , EN	https://www.chanel.com/sa/	Done
Korea,	https://www.chanel.com/kr/	Done
Kuwait, EN	https://www.chanel.com/kw/	Done
Latin America,	https://www.chanel.com/lx/	Done
Latvia , EN	https://www.chanel.com/lv/	Done
Lithuania , EN	https://www.chanel.com/lt/	Done
Luxembourg, EN	https://www.chanel.com/lu-fr/	Done
Malaysia, EN	https://www.chanel.com/my/	Done
Mexico	https://www.chanel.com/mx/	Done
Netherlands - English (UK)	https://www.chanel.com/nl/	Done
Norway,EN	https://www.chanel.com/no/	Done
Poland, EN	https://www.chanel.com/pl/	Done
Portugal , EN	https://www.chanel.com/pt/	Done
Qatar , EN	https://www.chanel.com/qa/	Done
Romania , EN	https://www.chanel.com/ro/	Done
Russia, EN	https://www.chanel.com/ru/	Done
Serbia, EN	https://www.chanel.com/rs/	Done
Singapore , EN	https://www.chanel.com/sg/	Done
Slovakia , EN	https://www.chanel.com/sk/	Done
Slovenia , EN	https://www.chanel.com/si/	Done
Spain , EN	https://www.chanel.com/es/	Done
Sweden , EN	https://www.chanel.com/se/	Done
Taiwan Region ,	https://www.chanel.com/tw/	Done

Thailand	https://www.chanel.com/th/	Done
Turkey , EN	https://www.chanel.com/tr/	Done
United Arab Emirate , EN Middle East (EN)	https://www.chanel.com/ae/ https://www.chanel.com/ae/	Done
UK , EN	https://www.chanel.com/gb/	Done
US , EN	https://www.chanel.com/us/	Done
Vietnam	https://www.chanel.com/vn	Done



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 - chnl.py

Here we are scraping the data as per the requirements

Step 1 - Importing required libraries

```
import json
import scrapy
from ..items import ChanelItem
from datetime import datetime
```

Step 2 -Here we created the spider "ChnlSpider" and added allowed domain, start url of all the product website URL are defined that we are crawling

```
    class ChnlSpider(scrapy.Spider):
    name = 'chanel'
    site = 'https://www.chanel.com'
    start_urls = ['https://www.chanel.com/us/']
    frag_url = 'https://www.chanel.com/us/fragrance/women/c/7x1x1/page-'
    eyegls_url = 'https://www.chanel.com/us/eyewear/eyeglasses/c/2x1x2/page-'
    sungls_url = 'https://www.chanel.com/us/eyewear/sunglasses/c/2x1x1/page-'
    watch_url = 'https://www.chanel.com/us/watches/collection/c/4x2/page-'
    fine_jewellery = 'https://www.chanel.com/us/fine-jewelry/collection/c/3x2/page-'
    face_url = 'https://www.chanel.com/us/makeup/face/c/5x1x6/page-'
    eye_url = 'https://www.chanel.com/us/makeup/eyes/c/5x1x4/page-'
    lips_url = 'https://www.chanel.com/us/makeup/lips/c/5x1x1/page-'
    nails_url = 'https://www.chanel.com/us/makeup/nails/c/5x1x7/page-'
    brushes_url = 'https://www.chanel.com/us/makeup/brushes-and-accessories/c/5x1x3/page-'
    nxp = []
    execution_id = '621291'
    feed_code = 'aeid5473'
    record_create_by = 'aeid5473_chanel'
    record_create_date = datetime.now()
    source_country = 'USA'
    for i in range(10):
        nxp.append(1)
```

Step 3 - 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 4 - Here we are fetching all links from selector and filtering the required urls

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

Step 5 - Here we are defining the functions for crawling the products category individually

```
  def sunglass(self , response):
  def eyeglass(self , response):
  def fragrance(self , response):
  def watches(self , response):
  def fine_jwlry(self , response):
  def mkp_face(self , response):
  def mkp_eye(self , response):
  def mkp_lips(self , response):
  def mkp_nails(self , response):
  def mkp_brushes(self , response):
```

Step 6 - For fashion category using multiple functions to access more than one page

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

Step 7 - Accessing the page to find products

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

Step 9 - Getting url to crawl products finally

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

Step 10 - Defining the function to crawl the required products

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

Step 11 - 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. **Color (Required) -** This we are getting from website
6. **Description (Required) -** This we are getting from website
7. **Material (Required) -** This we are getting from the website.
8. **Name (Required) -** This we are getting from the website.
9. **Price (Required) -** This we are getting from the website.
10. **product_id (Required) -** This we are getting from the website.
11. **Size (Required) -** This we are getting from the website.
12. **Record_create_by (Mandatory) -** This is hardcoded with spider name
13. **Record_create_dt (Mandatory) -** This is the timestamp for capturing the data.
14. **Site (Mandatory)-** This is hardcoded.
15. **Source (Mandatory) -** This is the link of the individual product.
16. **Source_country (Mandatory) -**This is hardcoded as per the specific country.

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.