eBay – Giant Bike Prices

Technical Design Document

Date – 14 Oct 2022

Table of Contents

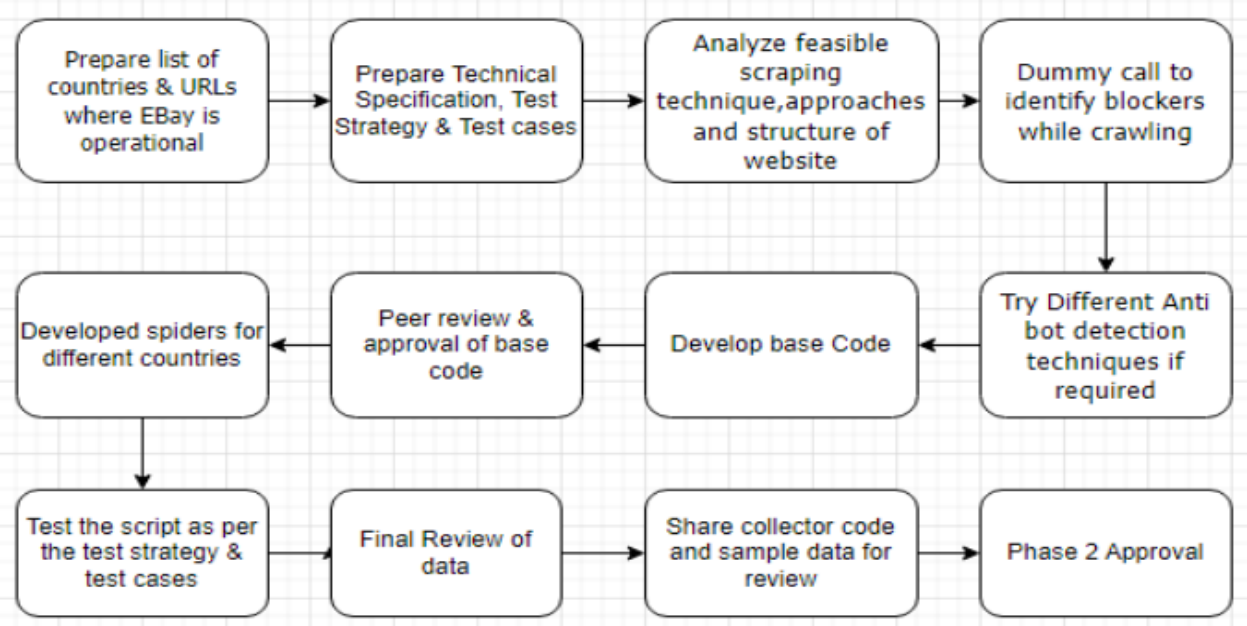
1. Scope of work

Get all listings (in particular, price) for all Giant bikes across as many countries as possible for eBay https://www.ebay.com/sch/i.html?_nkw=giant+bike&_sop=12

2. Solution Approach

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

- Prepared the below list of countries where EBay is operational.
- All the countries have the same search request and the same rendered layout.
- In some countries, the brand "Giant" is not listed so we have not created spider for the countries.
- For some products, we are getting different currency for price on the listing page as compared to the product detail page for the same product.
- In some countries, where the language is different from English, we are using language specific parameters to customize the code to capture the required value.



No. of countries where EBay is Operational	No. of Spiders created
19	15

Country Name	URL
Italy	https://www.ebay.it/sch/177831/i.html?_from=R40&_nkw=giant+Biciclette&_sop=10&_ipg=240 Website Operational - Yes Giant (Brand) is listed - Yes Category - Biciclette Spider Created - Yes
US	https://www.ebay.com/sch/i.html?_from=R40&_sacat=0&LH_TitleDesc=0&_nkw=giant+bicycles&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Bicycles Spider Created - Yes
UK	https://www.ebay.co.uk/sch/i.html?_from=R40&_nkw=giant+bikes&_sacat=0&_ipg=240&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Bikes Spider Created - Yes
Australia	https://www.ebay.com.au/sch/i.html?_from=R40&_nkw=giant+bike&_sacat=0&_ipg=240 Website Operational - Yes Giant (Brand) is listed - Yes Category - Bicycles Spider Created - Yes
Austria	https://www.ebay.at/sch/i.html?_from=R40&_nkw=Giant+Fahrr%C3%A4der&_sacat=0&_sop=10&_ipg=240

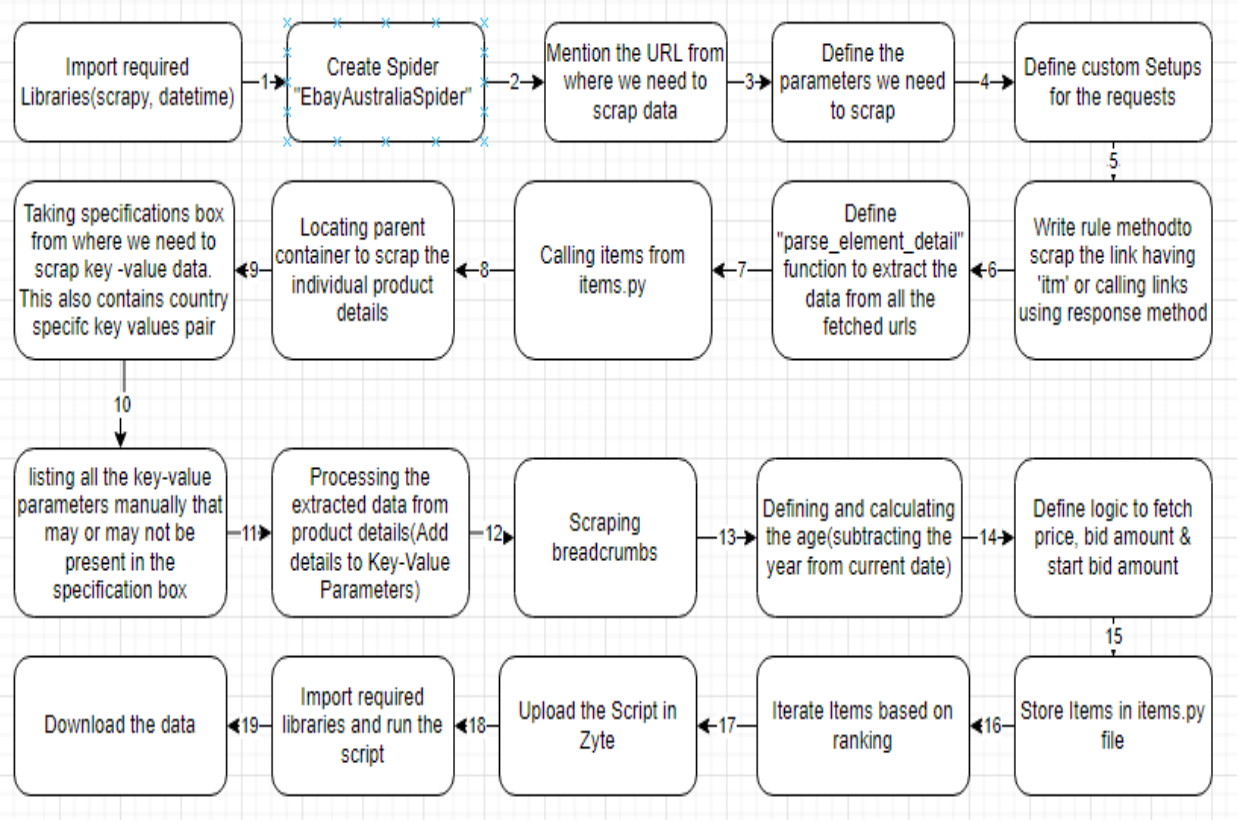
	Website Operational - Yes Giant (Brand) is listed - Yes Category - Bicycles Spider Created - Yes
Belgium	https://www.benl.ebay.be/sch/177831/i.html?_from=R40&_nkw=giant+Fietsen&LH_TitleDesc=0&_ipg=240&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Fietsen Spider Created - Yes
France	https://www.ebay.fr/sch/177831/i.html?_from=R40&_nkw=giant+v%C3%A9los&LH_TitleDesc=0&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Vélos Spider Created - Yes
Germany	https://www.ebay.de/sch/i.html?_from=R40&_nkw=giant+Fahrr%C3%A4der&_sacat=0&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Fahrräder Spider Created - Yes
Netherland	https://www.ebay.nl/sch/177831/i.html?_from=R40&_nkw=giant+Fietsen&_sop=10&_ipg=240 Website Operational - Yes Giant (Brand) is listed - Yes Category - Fietsen Spider Created - Yes
Spain	https://www.ebay.es/sch/177831/i.html?_from=R40&_nkw=giant+bicicletas&LH_TitleDesc=0&_sop=10&_ipg=240 Website Operational - Yes

	Giant (Brand) is listed - Yes Category - Bicicletas Spider Created - Yes
Switzerland	https://www.ebay.ch/sch/i.html?_from=R40&_nkw=giant+Fahrr%C3%A4der&_sacat=0&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Fahrräder Spider Created - Yes
Hong kong	https://www.ebay.com.hk/sch/177831/i.html?_from=R40&_nkw=Giant+bike&LH_TitleDesc=0&_ipg=240&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Bicycles Spider Created - Yes Data Scrapped - 205 Remarks - For some products, we are getting HK currency for price in listing page but in Euro's in product detail page
Ireland	https://www.ebay.ie/sch/i.html?_from=R40&_nkw=giant+bicycles&_sacat=0&LH_TitleDesc=0&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Bikes Spider Created - Yes
Malaysia	https://www.ebay.com.my/sch/i.html?_from=R40&_nkw=Giant+bike&_sacat=0&_sop=10 Website Operational - Yes Giant (Brand) is listed - No Spider Created - No, as the brand is not listed
Philippines	https://www.ebay.ph/sch/i.html?_from=R40&_nkw=Giant+bike&_sacat=0&_sop=10 Website Operational - Yes Giant (Brand) is listed - No

	Spider Created - No
Poland	https://www.ebay.pl/sch/177831/i.html?_from=R40&_nkw=giant+Rowery&LH_TitleDesc=0&_ipg=240&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Bikes Spider Created - Yes
Russia	https://www.ebay.com/sch/i.html?_from=R40&_trksid=p2380057.m570.l1313&_nkw=giant+bikes&_sacat=0 Website Operational - No Giant (Brand) is listed - NA Spider Created - No
Singapore	https://www.ebay.com.sg/sch/i.html?_from=R40&_nkw=giant+bikes&_sacat=0&_ipg=240&_sop=10 Website Operational - Yes Giant (Brand) is listed - Yes Category - Bicycles Spider Created - Yes Remarks - For some products, we are getting Singapore Dollar currency for price in listing page but in US Dollar on the product detail page.
Sweden	https://www.ebay.com/sch/i.html?_from=R40&_trksid=p3671980.m570.l1313&_nkw=giant+bikes&_sacat=0 Website Operational - No Giant (Brand) is listed - NA Spider Created - No

3. Script Development Flow

Below steps are followed to create each spider per term for each country.



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 - ebay_australia.py (Base code -Australia)

Here we are scraping Giant bikes(Australia) important and specifications data.

Step 1 - We have Imported Different bLibraries here that we have to use in our codes

```
import scrapy
import datetime

from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
from ..items import EbayItem
```

Step 2 - Here we have created one spider named "EbrayAustraliaSpider" that we have to deploy in zyte

```
class EbayAustraliaSpider(CrawlSpider):
    name = 'ebay_australia'
```

Step 3 - Here we have defined our EBay website url from where we are scraping the data

```
    start_urls =
["https://www.ebay.com.au/sch/i.html?_from=R40&_nkw=Giant+bike&_sacat=0&_ipg=240"]
```

Step 4 - Here we have defined mandatory data required as per the template

```
    type = "Bike"
    site = 'www.ebay.com.au'
    source_country = 'AU'  - Country specific parameter
    context_identifier = ' '
    item_ranking = 1
    record_created_by = " "
    execution_id = " "
          # Execution id will be taken automatically from zyte, for now this is hardcoded as we
have not implemented pipelines
    feed_code = "AEID-5183"
```

Step 5 - Here we have defined custom settings for Crawling data from the website

```
   custom_settings = {
                'ROBOTSTXT_OBEY': False,
```

```
'CONCURRENT_REQUESTS': 20,
'COOKIES_ENABLED': True,
'COOKIES_DEBUG': True,
'CONCURRENT_REQUESTS_PER_DOMAIN': 500,
'DOWNLOAD_DELAY': 0,
'AUTOTHROTTLE_ENABLED': False,
'DOWNLOAD_TIMEOUT': 20,
'DUPEFILTER_DEBUG': True,
}
```

Step 6 - Here rule method is applied, this is one of the method that is applied to crawl all the links containing "itm"

```
rules = (
    Rule(LinkExtractor(restrict_css='.pagination__next', deny=['_pgn=1','rt=nc'])),
    Rule(LinkExtractor(restrict_css='a.s-item__link'), callback='parse_element_detail'),
 )
```

Step 7 - Here we are defining "parse_element_detail" Function to extract the data from all the fetched urls

```
def parse_element_detail(self, response):          # Function to extract data from fetched urls
```

Step 8 - We are importing items here from items.py file

```
item = EbayItem()              # object to store data in "items.py"
```

Here we are taking parent container so that it will become easy to capture all the elements inside the container

```
element = (response.css("#LeftSummaryPanel .ux-textspans--BOLD::text").get())
```

We are taking breadcrumb data from here under try catch block to handle any exception

```
try :
    self.context_identifier = (response.css("li a.seo-breadcrumb-text span::text").get()) + ">"
+ (response.css("a[aria-current='location'] span::text").get())
except:
    self.context_identifier = (response.css("a[aria-current='location'] span::text").get())
```

Step 9 - Here we are taking the specifications box from where the key -value data needs to be scraped

```
specification_u = response.css(".ux-layout-section--features .ux-textspans::text").extract()
```

Step 10 - Country Specific Codes starts here

- Defining Key - Value pair as per the specific country

key_value = ["Condition:", "Model Year:", "Shifter Style:", "Handlebar Type:", "Number of Speeds:", "Color:", "Wheel Size:", "Suspension Type:", "Bike Type:", "Brand:",rame Size:", "Material:", "Department:", "Model:", "Brake Type:", "Tire Type:", "MPN:", "Vintage", "Gear Change Mechanism", "UPC", "Frame Number", "Features", "Item Weight", "Stem", "Saddle", "Drivetrain Type", "Seatpost", "Gender", "Crankset", "Chainrings","Cassette", "Wheel Set", "Front Derailleur", "Fork", "Rear Derailleur", "Shifters"]

key_value_lang = ["Zustand:", "Jahr:", "Shifter Style:", "Lenkerart:", "Number of Speeds:", "Farbe:", "Wheel Size:", "Federung:", "Fahrradart:", "Marke:","Rahmengröße:", "Material:", "Abteilung:", "Model:", "Bremsmethode:", "Reifentyp:", "MPN:", "Vintage", "Gear Change Mechanism", "UPC", "Frame Number","Features", "Item Weight", "Stem", "Saddle", "Drivetrain Type", "Seatpost", "Geschlecht", "Crankset", "Chainrings", "Cassette", "Wheel Set","Front Derailleur", "Fork", "Rear Derailleur", "Shifters"]

Step 11 - Here we are listing all the specifications key-value parameters manually that may or may not be present in the box

```
key_value = ["Condition:", "Model Year:", "Shifter Style:", "Handlebar Type:", "Number of Speeds:", "Color:", "Wheel Size:", "Suspension Type:", "Bike Type:", "Brand:","Frame Size:", "Material:", "Department:", "Model:", "Brake Type:", "Tire Type:", "MPN:", "Vintage", "Gear Change Mechanism", "UPC", "Frame Number","Features", "Item Weight", "Stem", "Saddle", "Drivetrain Type", "Seatpost", "Gender", "Crankset", "Chainrings","Cassette", "Wheel Set", "Front Derailleur", "Fork", "Rear Derailleur", "Shifters"]
```

Step 12 - Here we are defining **for loop** for key_value data in which we are replacing blank space with _ and blank space with

```
for key in key_value:
    item[key.replace(" ", "_").replace(":", "")] = ""
```

Step 13 - Here we are defining age manually for taking model year

```
item["Age"] = " "
```

Step 14 - Here we are defining one for loop for the length of specifications and inside that we are again defining one for loop for key value pair data-

```
for i in range(len(specification_u)-1):
    for j in key_value:
```

Defining 'if' for taking all specifications parameters from the box

```
            if j in specification_u[i]:
                item[j.replace(" ", "_").replace(":", "")] = specification_u[i + 1]
```

Defining 'if' for taking condition from the box

```
            if "Read more" in specification_u[i]:
                item["Condition"] = specification_u[i + 1]
```

Defining 'if' for taking model year from the box

```
            if "Model_Year" in specification_u[i]:

                date = datetime.date.today()
                current_year = date.year
                item["Age"] = int(current_year) - int(item["Model_Year"])
```

Here we are taking data that is defined in class using "self."

```
        self.record_created_by = self.name
        self.execution_id = "something"#environ.get('SHUB_JOBKEY', None)
```

Step 15 - Here we are storing items data in items.py

```
        item["record_create_dt"] = datetime.datetime.utcnow().strftime('%Y-%m-%d %T')
        item["type"] = self.type
```

Here it is taken from "self". because we have defined it in the class only

```
        item["element"] = element
        item["currency"] = "AU Dollar"
```

Here we have written logic for fetching price, bid amount and starting bid amount data

```
        if len(response.css("#prcIsum_bidPrice.notranslate::text").extract()) != 0 : #or
response.css("#prcIsum_bidPrice.notranslate::text").extract() != "" :

            item["starting_bid"] = response.css("#prcIsum_bidPrice.notranslate::text").extract()
            item["total_bid"] = response.css("#qty-test::text").get()
            item["price"] = ""

        elif response.css("#prcIsum::text").extract() !=0:

            item["price"] = response.css("#prcIsum::text").extract()
            item["starting_bid"] = ""
            item["total_bid"] = ""

        else:

            item["price"] = response.css("# mm-saleDscPrc::text").extract()
            item["starting_bid"] = ""
            item["total_bid"] = ""
```

Here we are storing items data in items .py and it is taken from self. because we have defined it in the class only

```
        item["feed_code"] = self.feed_code
        item["site"] = self.site
        item["source_country"] = self.source_country
```

```
item["context_identifier"] = self.context_identifier
item["record_create_by"] = self.record_created_by
item["item_ranking"] = self.item_ranking
item["src"] = response
yield item
```

Step 16 - For iterating item ranking defined in class

```
self.item_ranking += 1
```

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. **key -** Zyte by default add this as an identifier.
2. **Age (Required) -** If the user mentions the model year then we are picking the value and calculating the age
3. **Bike_Type -** We are getting this from a parameter.
4. **Brake_Type -** We are getting this from a parameter.
5. **Brand -** We are getting this from a parameter.
6. **Cassette -** We are getting this from a parameter.
7. **Chainrings -** We are getting this from a parameter.
8. **Color -** We are getting this from a parameter.
9. **Condition (Required) -** We are getting this from a parameter.
10. **Crankset -** We are getting this from a parameter.
11. **Department -** We are getting this from a parameter.
12. **Drivetrain_Type -** We are getting this from a parameter.
13. **Features -** We are getting this from a parameter.
14. **Fork -** We are getting this from a parameter.
15. **Frame_Number -** We are getting this from a parameter.
16. **Front_Derailleur -** We are getting this from a parameter.
17. **Gear_Change_Mechanism -** We are getting this from a parameter.
18. **Gender -** We are getting this from a parameter.
19. **Handlebar_Type -** We are getting this from a parameter.
20. **Item_Weight -** We are getting this from a parameter.
21. **MPN -** We are getting this from a parameter.
22. **Material -** We are getting this from a parameter.
23. **Model -** We are getting this from a parameter.
24. **Model_Year -** We are getting this from a parameter.
25. **Number_of_Speeds -** We are getting this from a parameter.
26. **Rear_Derailleur -** We are getting this from a parameter.
27. **Saddle -** We are getting this from a parameter.
28. **Seatpost -** We are getting this from a parameter.
29. **Shifter_Style -** We are getting this from a parameter.
30. **Shifters -** We are getting this from a parameter.

31. **Stem -** We are getting this from a parameter.
32. **Suspension_Type -** We are getting this from a parameter.
33. **Tire_Type -** We are getting this from a parameter.
34. **UPC -** We are getting this from a parameter.
35. **Vintage -** We are getting this from a parameter.
36. **Wheel_Set -** We are getting this from a parameter.
37. **Wheel_Size -** We are getting this from a parameter.
38. **context_identifier (Mandatory) -** Here we are capturing the breadcrumbs to identify the product category.
39. **currency -** We are getting this from a parameter.
40. **execution_id (Mandatory) -** Execution id will be taken automatically from zyte, for now this is hardcoded as we are not using pipelines.
41. **feed_code (Mandatory) -** This is hardcoded as we have not any pro
42. **item_ranking (Required) -**We are getting this from a parameter while searching from newly listed criteria.
43. **price (Required) -** We are getting this from a parameter.
44. **record_create_by (Mandatory) -** This is hardcoded with spider name
45. **record_create_dt (Mandatory) -** This is the timestamp for capturing the data.
46. **site (Mandatory)-** This is hardcoded.
47. **source_country (Mandatory) -**This is hardcoded as per the project code.
48. **src (Mandatory) -** This is the link for product details page.
49. **starting_bid (Required) -** We are getting this from a parameter.
50. **title (Mandatory) -** We are getting this from a parameter.
51. **total_bid (Required) -** We are getting this from a parameter.
52. **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.
Once the products are listed, there are chances they can be delisted from the website once the bid is closed or the product is sold out.	The data will keep on changing for the same product based on the status of listing.