ext_aeid5234-Ask - XPENG Inventory Tracking

Analysis & Design Document

Date – 28 Oct 2022

Table of Contents

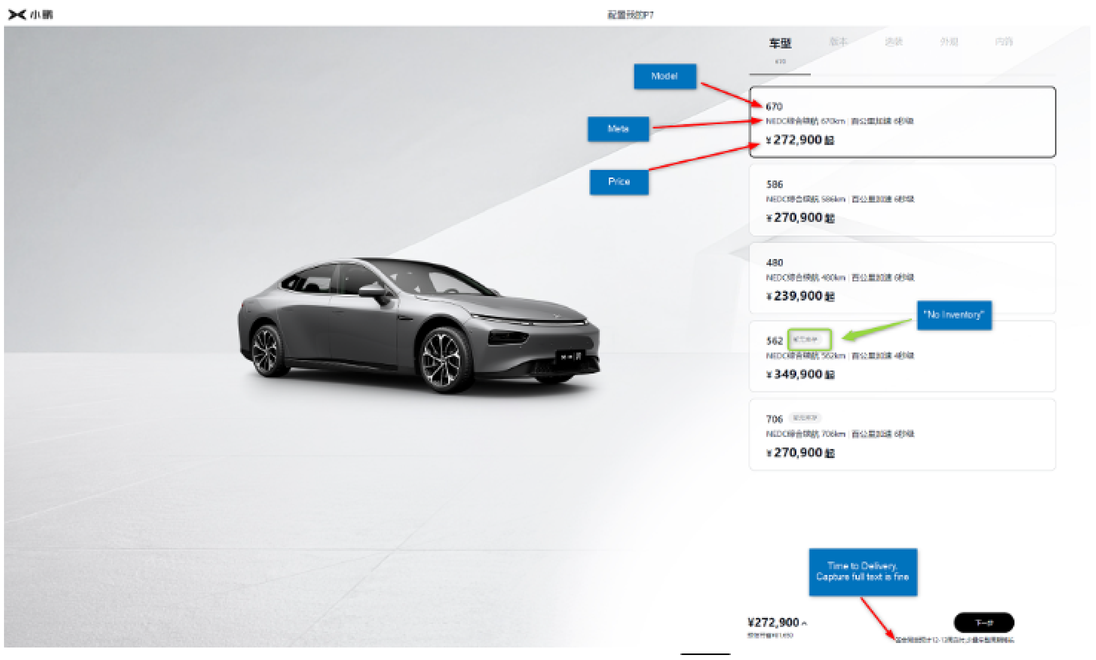
1. Scope of work

For SITE: https://store.xiaopeng.com/carDeploy.html?entry=12_1_2#/P7/step1

Collect the below highlighted data in screenshots for each model,

1. Capture price
2. Meta data
3. Time-to-Delivery

Note: There are models which do not have inventory.



670
NEDC综合续航 670km | 百公里加速 6秒级
¥272,900 起

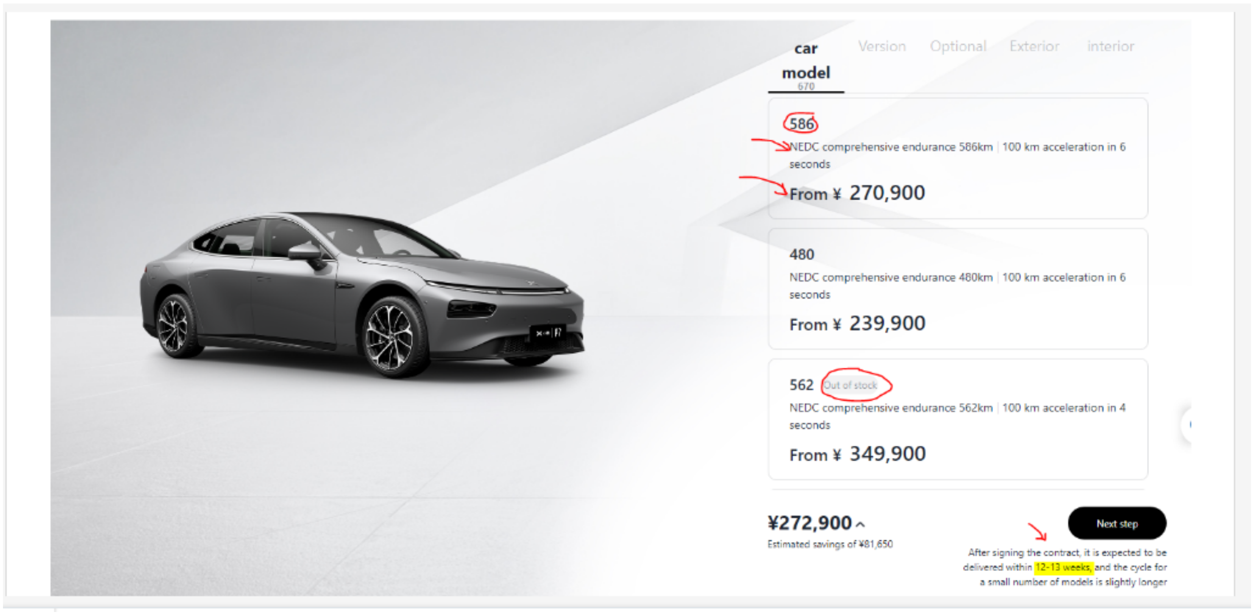
586
NEDC综合续航 586km | 百公里加速 6秒级
¥270,900 起

480
NEDC综合续航 480km | 百公里加速 6秒级
¥239,900 起

562
NEDC综合续航 562km | 百公里加速 4秒级
¥349,900 起

706
NEDC综合续航 706km | 百公里加速 6秒级
¥270,900 起

¥272,900



586
NEDC comprehensive endurance 586km | 100 km acceleration in 6 seconds
From ¥ 270,900

480
NEDC comprehensive endurance 480km | 100 km acceleration in 6 seconds
From ¥ 239,900

562 Out of stock
NEDC comprehensive endurance 562km | 100 km acceleration in 4 seconds
From ¥ 349,900

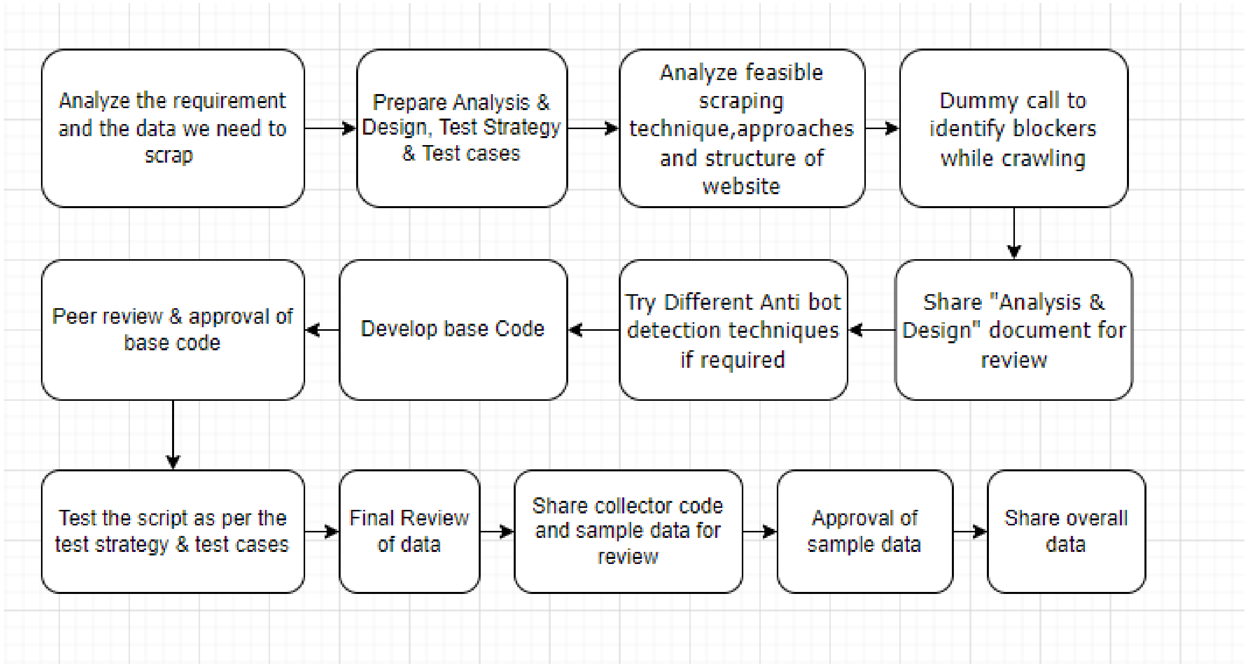
¥272,900
Estimated savings of ¥81,650

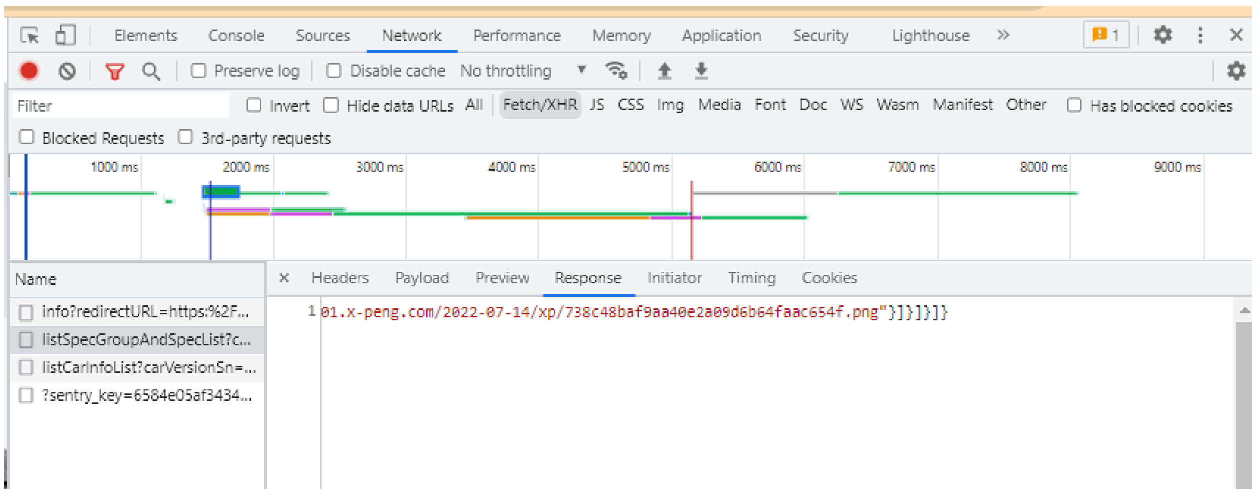
After signing the contract, it is expected to be delivered within 12-13 weeks, and the cycle for a small number of models is slightly longer

2. Solution Approach

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

- The website is **global**, hence only one collector code is needed.
- 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 getting the data when we are checking the fetch/XHR via Network.
- We are using request function to capture the data.


Analyze the requirement and the data we need to scrap
Prepare Analysis & Design, Test Strategy & Test cases
Analyze feasible scraping technique,approaches and structure of website
Dummy call to identify blockers while crawling
Share "Analysis & Design" document for review
Try Different Anti bot detection techniques if required
Develop base Code
Peer review & approval of base code
Test the script as per the test strategy & test cases
Final Review of data
Share collector code and sample data for review
Approval of sample data
Share overall data

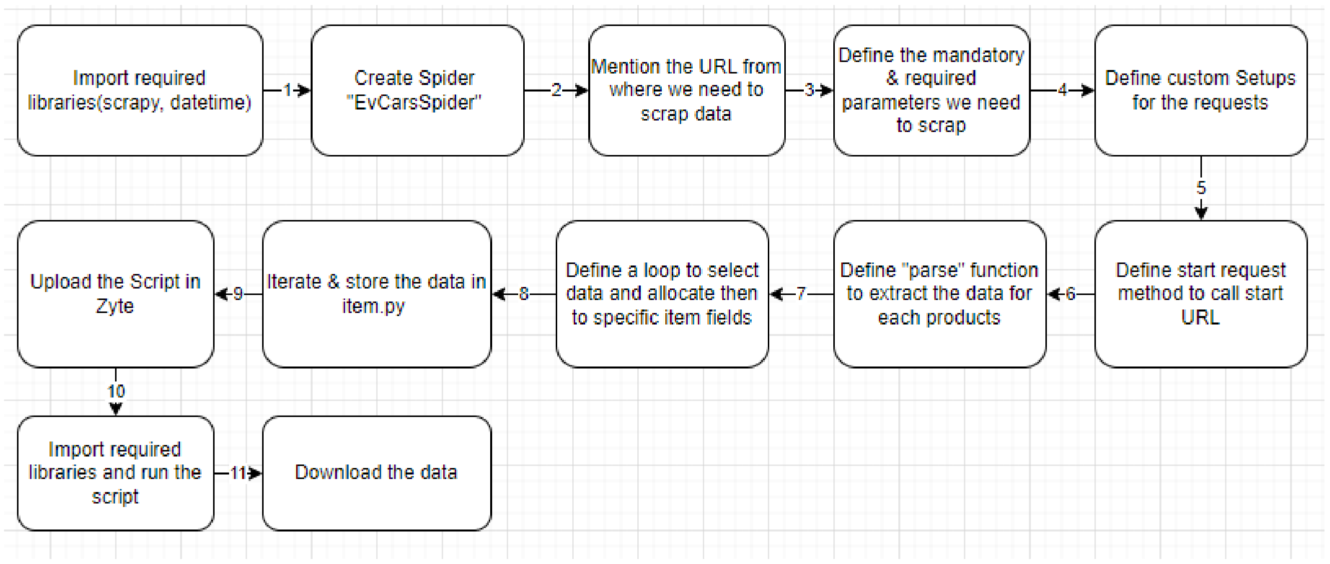


```
info?redirectURL=https%2F...
listSpecGroupAndSpecList?c...
listCarInfoList?carVersionSn=...
?sentry_key=6584e05af3434...
```

```
01.x-peng.com/2022-07-14/xp/738c48baf9aa40e2a09d6b64faac654f.png"}]}]}]}
```

3. Script Development Flow

Below steps are followed to create spider


Import required libraries(scrapy, datetime)
1
Create Spider "EvCarsSpider"
2
Mention the URL from where we need to scrap data
3
Define the mandatory & required parameters we need to scrap
4
Define custom Setups for the requests
5
Define start request method to call start URL
6
Define "parse" function to extract the data for each products
7
Define a loop to select data and allocate then to specific item fields
8
Iterate & store the data in item.py
9
Upload the Script in Zyte
10
Import required libraries and run the script
11
Download the data

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 - ev_cars.py

Here we are scraping the required and mandatory data

Step 1 - Importing required libraries here

```
import scrapy
import json
import datetime
from os import environ
from ..items import XiaopengItem
```

Step 2 - Here a spider named "EvCarsSpider" is created

```
class EvCarsSpider(scrapy.Spider):
    name = 'ev_cars'
```

Step 3 - Here allowed domain and start url of all the 4 cars websites are defined that we are crawling

```
    allowed_domains = ['http://store.xiaopeng.com/']
    start_urls =
['https://store.xiaopeng.com/configurate.html?carSeries=P7&entry=12_1_2#/P7/step1',
"https://store.xiaopeng.com/configurate.html?carSeries=P5&entry=12_1_2#/P5/step1",
"https://store.xiaopeng.com/configurate.html?carSeries=G3i&entry=12_1_2#/G3i/step1",
"https://store.xiaopeng.com/configurate.html?carSeries=G9&entry=12_1_2#/G9/step1"]
```

Step 4 - Here all Mandatory Fields Data are defined under the main class that will be called using "self".

```
    execution_id = ""  # This will be taken automatically from zyte
    feed_code = "aeid5234"
    record_create_by = "aeid5234_ev_cars"
    record_create_dt = datetime.datetime.utcnow().strftime('%Y-%m-%d %T')
    site = "https://store.xiaopeng.com"
    source_country = "Global"
    src = ""
    type = "XPENG EV CARS"
```

Step 5 - Here we are defining custom settings that are needed for crawling.

```
custom_settings = {
     'ROBOTSTXT_OBEY': False,
     'COOKIES_ENABLED': True,
     'COOKIES_DEBUG': True,
     'AUTOTHROTTLE_ENABLED': True,
     'DOWNLOAD_TIMEOUT': 20,
     'DUPEFILTER_DEBUG': True,
  }
```

Step 6 - Here we are defining start_requests function for starting the crawling requests of the urls.

```
def parse(self, response):
  items = XiaopengItem()                    # Object to store data in items.py
  data = response.css('script::text').get()      # Here we are fetching script data in which all
the website page data is present
```

Step 7 - Here we are Removing extra data coming at front and end of the script

```
data = data.replace('window.__INITIAL_STATE__=', "")
data = data.replace(";", "")
```

Step 8 - Here we are converting the coming script data from string to dictionary using json.loads

```
data1 = json.loads(data)
```

Step 9 - Here from this for loop we are fetching all the data that is required from the website by going one by one inside dictionaries

```
for i in data1['vgroups']:
   #Here we are checking the "carSeriesCode" and proceeding accordingly
   if (i['carSeriesCode'] == 'P7') or (i['carSeriesCode'] == 'P5') or (i['carSeriesCode'] ==
'G3i') or (i['carSeriesCode'] == 'G9'):
```

Step 10 - Here we are fetching required data which is available like Model, Inventory, Car series name in this and for other required value which are not present we are passing empty string

```
items["Model"] = i['carYearName']          # For fetching car model
items["Car"] = i['carSeriesName']          # For fetching car series name
```

```
        items["Car_Version"] = ""
        items["Delivery_time"] = ""
        items["Price"] = ""
```

Step 11 - Here we are using for loop for iterating all the required data like car model, Car series name, Car versions, Meta data, Price, No inventory, Delivery time

```
    for j in i['carVersionList']:
```

Step 12 - Here we are storing mandatory data in items.py and some data is taken from self. because we have defined it in the main class

```
        items['Execution_id'] = environ.get('SHUB_JOBKEY', None)
        items["Feed_code"] = self.feed_code
        items["Record_create_by"] = self.record_create_by
        items["Record_create_dt"] = self.record_create_dt
        items["Site"] = self.site
        items["Source_country"] = self.source_country
        items["Src"] = self.src
        items["Type"] = self.type
        items["Src"] = response.url
        yield items                          # yielding all items here
```

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. **Car (Required) -** We are capturing the Car type.
3. **Car_Version (Required) -** We are capturing the car version**.**
4. **Delivery_time (Required) -** We are capturing the expected delivery time for the car.
5. **Execution_id (Mandatory) -** Execution id will be taken automatically from zyte.
6. **Feed_code (Mandatory) -** This is hardcoded as project name.
7. **Inventory (Required) -** This will have month and year data.
8. **Metadata (Required) -** We are capturing the product details.
9. **Model (Required) -** We are capturing the model name.
10. **Price (Required) -** We are capturing the price of each model.
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_country (Mandatory) -**This is hardcoded as the website is global.
15. **Src (Mandatory) -** This is the link for product details page.
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.