DXP-Product Bot

1. Overview

The project's aim is to validate all product categories using **UPC** and **Store codes**, ensuring the complete accuracy of data displayed on the Frontend, aligning with the information stored in the database.

With a large quantity of products, approximately over a **million rows** processed **daily** through ETL, it is necessary to employ a technique to run the verification program comprehensively and complete it within a reasonable timeframe. This ensures that product information is accurate, allowing online shoppers to make purchases without any issues.

Based on that, it is necessary to write an endpoint to run the code daily after the ETL process is complete for verification. The API is written using the Python framework **FastAPI**.

- Data sample of product

```
data = {
    'Products': [
        {
            'Id': '00020615300000',
            'Name': 'FRESH WILD CAUGHT Swordfish Steak',
            'ShortDescription': 'Cut to Order',
            'FullDescription': '',
            'CompanyId': None,
            'ProductType': None,
            'Price': 17.99,
            'ShowMainPrice': None,
            'CostPrice': None,
            'SalePrice': 10.99,
            'SalePriceText': '$10.99',
            'ShowSalePrice': None,
            'TaxClass': 'Default Tax Class',
            'AvailableInAllStore': True,
            'AvailableInStore': True,
            'Available': True,
            'InStock': True,
            'DefaultImage': 'https://../marketing/Product_Art_UPC/00206153000007.jpg',
            'WarrantyInfo': None,
            'Tags': None,
            'Condition': None,
            'DisplayOrder': None,
            'UPC': '00020615300000',
            'MinPurchaseQuantity': None,
            'MaxPurchaseQuantity': None,
            'ValueType': None,
            'ItemSize': None,
            'OnSale': True,
            'InventoryTracking': None,
            'CurrentStock': None,
            'LowStock': None,
            'Information': None,
            'ProductVariants': [
                {
                    'Id': None,
                    'ProductId': '00020615300000',
                    'ProductName': 'FRESH WILD CAUGHT Swordfish Steak',
                    'Name': 'Cut to Order',
                    'Description': 'Cut to Order',
                    'Sku': '00020615300000'
                }
            ],
            'ProductAttributes': [],
            'ProductCategories': [
                {
                    'CategoryId': '11986',
                    'CategoryName': 'Seafood'
                }
            ],
            'PriceText': '$17.99',
            'SaleInfo': None,
            'Category': 'Seafood',
            'ExternalId': None,
            'Sizes': None,
            'ItemKey': '00020615300000',
            'UnitPrice': '1 LB  <br> <span class="sale-note">Price with membership</span>',
            'Disclaimer': '',
            'Ingredients': None,
            'DefaultCategory': None,
            'WeightIncrement': {
                'Abbreviation': '1 LB ',
                'Type': None,
                'Label': None,
                'Size': 1.25,
                'MaxSize': 999,
                'MinSize': 0,
                'Range': None
            },
            'Promotions': []
        }
    ],
    'DeptSummary': [
        {
            'DepartmentId': '11986',
            'Department': 'Seafood',
            'Count': None,
            'SubDept1s': []
        }
    ],
    'AttributeSummary': [],
    'BrandSummary': [
        {
            'BrandId': 'FRESH WILD CAUGHT',
            'BrandName': 'FRESH WILD CAUGHT',
            'Count': 1
        }
    ],
    'TotalOnSaleProducts': 0,
    'TotalPages': 1,
    'TotalRecords': 1,
    'PageSize': 1,
    'PageIndex': 1
}
```

- Required fields to verify

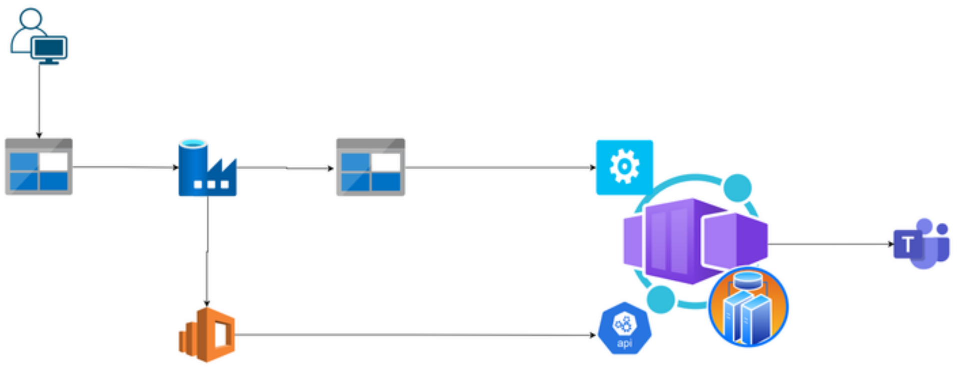
UPC	STORE_CODE	PRICE	NAME	CATEGORY	DESCRIPTION	IMAGE

From the **UPC** and **Store codes**, we proceed to retrieve information for the mentioned fields by calling an API to return the data.

- API to get data

```
https://../scrsecom/v2.0/api/Product/search
```

2. Workflow



workflow

- Source file: **price_sale_of_product.csv** and **price_regular_of_product.csv**
- Destination file: **price_sale_of_product_check.csv** and **price_regular_of_product_check.csv**

Both the input and output files are located within the same container.

- Implementation workload: The bot is run on a server that has been configured and deployed with code on Kubernetes (K8s), along with Jenkins and Docker. Continuous Integration/Continuous Deployment (CI/CD) processes are set up to trigger when code is committed to Git.
- Notification: Check notifications in the webhook after the execution is complete.

3. Conclusion

To manually excute this bot, we could use this endpoint

```
https://../product_sale_check/
```

```
https://../product_regular_check/
```