# **Egyptian HS Code Tariff Scraper - Technical Documentation**

Ogbonna Ngwu

## **1. Overview**

This Python script automates the extraction of Egyptian customs tariff data (Import Taxes and VAT) from the [Nafeza government portal](https://www.nafeza.gov.eg/) for a comprehensive list of HS codes. It handles over 500 HS codes, extracts tax rates, and exports structured data to Excel.

## 

## 

## 

## 

## 

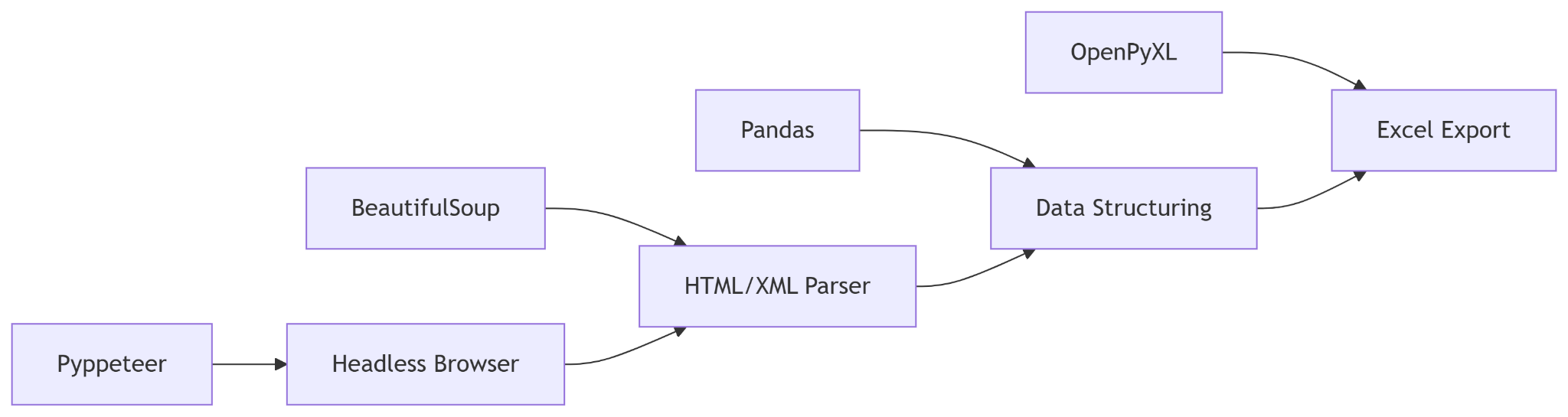
## 

## 

## 

## **2. System Architecture**

### **2.1 Component Diagram**



### 

### **2.2 Technology Stack**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| Browser Automation | Pyppeteer | Simulates human browsing behavior |
| HTML Parsing | BeautifulSoup | Extracts data from API responses |
| Data Handling | Pandas | Structures scraped data |
| Excel Export | OpenPyXL | Generates formatted spreadsheets |
| Async Management | asyncio | Handles concurrent operations |

## **3. Code Walkthrough**

### **3.1 Core Functions**

#### scrape\_hs\_code(hs\_code, browser)

**Purpose**: Handles single HS code scraping with error encapsulation.

#### parse\_response(content)

**Key Logic**:

* Uses Arabic keywords to identify tax types
* Returns structured dictionary with fallback "N/A" values

#### get\_browser()

**Fallback Mechanism**:

1. Attempts to use local Chrome
2. Downloads Chromium v818858 if missing

### **3.2 Main Execution Flow**

## **4. Data Structure**

### **4.1 Output Format**

| **Column** | **Type** | **Description** | **Example** |
| --- | --- | --- | --- |
| HS\_Code | String | 10-digit HS code | "0303920000" |
| Description | String | Arabic product description | "زعانف اسماك..." |
| Import\_Tax\_Basic | String | Base import duty rate | "20.000 %" |
| Import\_Tax\_EFTA | String | EFTA preferential rate | "2.500 %" |
| VAT | String | Value Added Tax | "14.000 %" |
| Status | String | Success/Error message | "Success" |

## **5. Error Handling**

### **5.1 Common Scenarios**

| **Error Type** | **Handling Method** | **Example** |
| --- | --- | --- |
| Connection Timeout | Retry up to 3 times | TimeoutError |
| Missing Data Fields | Populate with "N/A" | KeyError |
| Invalid Response Format | Skip record, log error | Malformed XML |

## **6. Performance Optimization**

### **6.1 Timing Benchmarks**

| **Operation** | **Avg. Time** | **Notes** |
| --- | --- | --- |
| Page Load | 2.1s | Depends on network latency |
| Data Extraction | 0.3s | Per HS code |
| Full 500-code Run | ~25 mins | With 3s random delays |

### **6.2 Memory Management**

* Each browser page closed after use (finally: await page.close())
* Chromium processes terminated post-execution