**Step 1: Set Up Your Environment**

1. **Install Python and required libraries** such as requests beautifulsoup4
2. **Set up MySQL**:
   * Create a new database for storing the scraped data.

### Step 2: Write the Scraping Script using Python

### Steps to extract data from urls such as

### Social Media Links

### Tech Stack (MVC, CMS, JS type etc)

### Meta Title

### Meta Description

### Payment Gateways (e.g., PayPal, Stripe, Razorpay)

### Website language

### Category of website

import requests

from bs4 import BeautifulSoup

import csv

# Function to extract meta tags

def extract\_meta(soup):

    meta\_title = soup.find('title').text if soup.find('title') else ''

    meta\_description = soup.find('meta', attrs={'name': 'description'})

    meta\_description = meta\_description['content'] if meta\_description else ''

    return meta\_title, meta\_description

# Function to extract social media links

def extract\_social\_media\_links(soup):

    social\_media\_links = []

    for a in soup.find\_all('a', href=True):

        if any(s in a['href'] for s in ['facebook.com', 'twitter.com', 'instagram.com', 'linkedin.com', 'youtube.com']):

            social\_media\_links.append(a['href'])

    return ', '.join(social\_media\_links)

# Function to extract tech stack

def extract\_tech\_stack(soup):

    scripts = soup.find\_all('script')

    tech\_stack = []

    for script in scripts:

        if 'jquery' in str(script).lower():

            tech\_stack.append('jQuery')

        if 'react' in str(script).lower():

            tech\_stack.append('React')

        if 'vue' in str(script).lower():

            tech\_stack.append('Vue.js')

        if 'angular' in str(script).lower():

            tech\_stack.append('Angular')

    return ', '.join(set(tech\_stack))

# Function to extract payment gateways

def extract\_payment\_gateways(soup):

    payment\_gateways = []

    if 'paypal' in soup.text.lower():

        payment\_gateways.append('PayPal')

    if 'stripe' in soup.text.lower():

        payment\_gateways.append('Stripe')

    if 'razorpay' in soup.text.lower():

        payment\_gateways.append('Razorpay')

    return ', '.join(payment\_gateways)

# Function to extract language

def extract\_language(soup):

    language = soup.find('html')['lang'] if soup.find('html') and 'lang' in soup.find('html').attrs else ''

    return language

# Function to determine category (dummy function)

def determine\_category(url):

    return 'General'

# Function to scrape a single website

def scrape\_website(url):

    try:

        response = requests.get(url)

        soup = BeautifulSoup(response.text, 'html.parser')

        meta\_title, meta\_description = extract\_meta(soup)

        social\_media\_links = extract\_social\_media\_links(soup)

        tech\_stack = extract\_tech\_stack(soup)

        payment\_gateways = extract\_payment\_gateways(soup)

        language = extract\_language(soup)

        category = determine\_category(url)

        return {

            'url': url,

            'meta\_title': meta\_title,

            'meta\_description': meta\_description,

            'social\_media\_links': social\_media\_links,

            'tech\_stack': tech\_stack,

            'payment\_gateways': payment\_gateways,

            'language': language,

            'category': category

        }

    except Exception as e:

        print(f"Failed to scrape {url}: {e}")

        return None

# Main function to scrape multiple websites from a CSV file

def main(csv\_file, output\_file):

    scraped\_data = []

    try:

        with open(csv\_file, newline='') as file:

            reader = csv.DictReader(file)

            for row in reader:

                url = row['url'].strip()

                if url:

                    data = scrape\_website(url)

                    if data:

                        scraped\_data.append(data)

        # Write scraped data to CSV file for SQL table creation

        with open(output\_file, 'w', newline='', encoding='utf-8') as csvfile:

            fieldnames = ['url', 'meta\_title', 'meta\_description', 'social\_media\_links', 'tech\_stack', 'payment\_gateways', 'language', 'category']

            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

            writer.writeheader()

            for data in scraped\_data:

                writer.writerow(data)

        print(f"Output saved to {output\_file}")

    except FileNotFoundError:

        print(f"CSV file '{csv\_file}' not found.")

if \_\_name\_\_ == "\_\_main\_\_":

    csv\_file = 'Book1.csv'

    output\_file = 'scraped\_data.csv'

    main(csv\_file, output\_file)

### Coded to make csv data UTF-8 Encoded:

import csv

def convert\_csv\_to\_utf8(input\_file, output\_file):

    # Read the original CSV file

    with open(input\_file, 'r', encoding='utf-8-sig') as infile:  # 'utf-8-sig' handles UTF-8 with BOM

        reader = csv.reader(infile)

        rows = list(reader)

    # Write to a new CSV file with UTF-8 encoding

    with open(output\_file, 'w', newline='', encoding='utf-8') as outfile:

        writer = csv.writer(outfile)

        writer.writerows(rows)

    print(f"File '{input\_file}' has been converted to UTF-8 and saved as '{output\_file}'")

if \_\_name\_\_ == "\_\_main\_\_":

    input\_file = 'scraped\_data.csv'  # Replace with your input file name

    output\_file = 'scraped\_data\_utf8.csv'  # Replace with your desired output file name

    convert\_csv\_to\_utf8(input\_file, output\_file)

### Step 3: Write SQL Script to Create Tables

Created an SQL script to set up database schema.

* Use NVARCHAR for all text columns to support Unicode characters.

CREATE TABLE Website (

Id INT IDENTITY(1,1) PRIMARY KEY,

Url NVARCHAR(MAX),

MetaTitle NVARCHAR(MAX),

MetaDescription NVARCHAR(MAX),

SocialMediaLinks NVARCHAR(MAX),

TechStack NVARCHAR(MAX),

PaymentGateways NVARCHAR(MAX),

Language NVARCHAR(MAX),

Category NVARCHAR(MAX)

);

**Step 3: Steps to upload data in SQL Server Management Studio (SSMS)**

1. **Open SQL Server Import and Export Wizard**:
   * Right-click on your database.
   * Go to **Tasks** > **Import Data**.
2. **Configure the Source**:
   * Select **Flat File Source**.
   * Browse to your scraped\_data.csv file.
   * Ensure **Unicode (UTF-8)** is selected as the file encoding.
   * Check the option for **Column names in the first data row**.
3. **Configure the Destination**:
   * Choose **SQL Server Native Client**.
   * Configure the connection to your database.
   * Select the Website table.
4. **Edit Mappings**:
   * Ensure each column is mapped correctly.
   * Adjust column lengths to NVARCHAR(MAX) if necessary.
   * Check for warnings about truncation and ensure all columns are set to the correct data types.
5. **Run the Import**:
   * Proceed through the wizard to complete the import.

**Challenges encountered: (See report given)**

Challenge has been seen while uploading data into sql database.

* Executing (Error) Messages Error 0xc02020a1: Data Flow Task 1: Data conversion failed. The data conversion for column "url" returned status value 4 and status text "Text was truncated or one or more characters had no match in the target code page.". (SQL Server Import and Export Wizard)
* Error 0xc020902a: Data Flow Task 1: The "Source - scraped\_data\_utf8 (1)\_csv.Outputs[Flat File Source Output].Columns[url]" failed because truncation occurred, and the truncation row disposition on "Source - scraped\_data\_utf8 (1)\_csv.Outputs[Flat File Source Output].Columns[url]" specifies failure on truncation. A truncation error occurred on the specified object of the specified component. (SQL Server Import and Export Wizard)
* Error 0xc0202092: Data Flow Task 1: An error occurred while processing file "C:\Users\SHILPA\Desktop\scraped\_data\_utf8 (1).csv" on data row 2. (SQL Server Import and Export Wizard)
* Error 0xc0047038: Data Flow Task 1: SSIS Error Code DTS\_E\_PRIMEOUTPUTFAILED. The PrimeOutput method on Source - scraped\_data\_utf8 (1)\_csv returned error code 0xC0202092. The component returned a failure code when the pipeline engine called PrimeOutput(). The meaning of the failure code is defined by the component, but the error is fatal and the pipeline stopped executing. There may be error messages posted before this with more information about the failure. (SQL Server Import and Export Wizard)

To cope up challenges:

**Steps to Rectify Truncation Error:**

1. **Check Column Width in SQL Server:**
   * Ensure that the meta\_title column in your Websites table in SQL Server is defined with sufficient width to accommodate the longest possible meta title you expect to import. For NVARCHAR, using MAX typically allows up to 2GB of data, but practical limits should be set based on your data expectations.
2. **Review Data in CSV File:**
   * Open your scraped\_data\_utf8\_final.csv file and examine the data in the meta\_title column. Check if there are unusually long strings that might exceed the expected length.
3. **Adjust Python Script (if necessary):**
   * Ensure that your Python script (scrape\_websites.py) handles data lengths appropriately before writing to the CSV file. If meta\_title or any other field might exceed practical limits, truncate or handle such cases programmatically.
4. **Ensure UTF-8 Encoding:**
   * Confirm that your CSV file (scraped\_data\_utf8\_final.csv) is saved with UTF-8 encoding to match the encoding specified in your Python script.
5. **Retry Import in SQL Server:**
   * After making adjustments, retry importing the scraped\_data\_utf8\_final.csv file into SQL Server using the SQL Server Import and Export Wizard.

### Step 4: Step3 not executed so Uploaded csv file manually to SQL database using online platform

### Step 5: Final documentation