Data Scientist Intern Assignment AllHeartWeb Inc.

Submitted by: Ritik Saini

To achieve the task of extracting information from a list of websites, we will follow these steps:

1. **Web Scraping**: Use Python with BeautifulSoup and Selenium to scrape the required information.
2. **Data Storage**: Store the extracted information in a MySQL database.
3. **Documentation**: Provide clear instructions for setup and running the solution.
4. **Code Quality**: Write well-organized, readable code following best practices.

Here's a detailed breakdown of each step:

Step 1: Web Scraping

We'll use Python for web scraping due to its rich ecosystem of libraries. We'll use `BeautifulSoup` for parsing HTML and `Selenium` for handling dynamic content.

Required Libraries

- `requests`
- `beautifulsoup4`
- `selenium`
- `mysql-connector-python`

Install these libraries using pip:

```
bash
Copy code
pip install requests beautifulsoup4 selenium mysql-connector-python
```

Setting Up Selenium

Download the appropriate WebDriver for your browser and ensure it's in your system's PATH. For example, for Chrome, download ChromeDriver from here.

Web Scraping Code

Here's a Python script for scraping the required information:

```
python
Copy code
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
import mysql.connector
import re

# Setup MySQL connection
```

```
db = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="web_info"
)
cursor = db.cursor()

# Create tables
cursor.execute("""
CREATE TABLE IF NOT EXISTS websites (
    id INT AUTO_INCREMENT PRIMARY KEY,
    url VARCHAR(255),
    meta_title VARCHAR(255),
    meta_description TEXT,
    tech_stack TEXT,
    payment_gateways TEXT,
    website_language VARCHAR(50),
    category VARCHAR(50),
    social_media_links TEXT
)
""")

def get_meta_info(soup):
    title = soup.find('title').text if soup.find('title') else ''
    description = ''
    if soup.find('meta', attrs={'name': 'description'}):
        description = soup.find('meta', attrs={'name':
'description'})['content']
    return title, description

def get_tech_stack(soup):
    tech_stack = []
    scripts = soup.find_all('script')
    for script in scripts:
        if 'react' in script.get('src', '').lower():
            tech_stack.append('React')
        if 'angular' in script.get('src', '').lower():
            tech_stack.append('Angular')
        if 'vue' in script.get('src', '').lower():
            tech_stack.append('Vue.js')
    return ', '.join(tech_stack)

def get_social_media_links(soup):
    social_links = []
    for a in soup.find_all('a', href=True):
        if 'facebook.com' in a['href']:
            social_links.append('Facebook: ' + a['href'])
        if 'twitter.com' in a['href']:
            social_links.append('Twitter: ' + a['href'])
        if 'linkedin.com' in a['href']:
            social_links.append('LinkedIn: ' + a['href'])
        if 'instagram.com' in a['href']:
            social_links.append('Instagram: ' + a['href'])
    return ', '.join(social_links)

def get_payment_gateways(soup):
    gateways = []
    if 'paypal' in soup.text.lower():
        gateways.append('PayPal')
    if 'stripe' in soup.text.lower():
        gateways.append('Stripe')
    if 'razorpay' in soup.text.lower():
        gateways.append('Razorpay')
    return ', '.join(gateways)

def get_website_language(soup):
    if soup.find('html'):
        return soup.find('html').get('lang', '')
    return ''

def categorize_website(soup):
    # This is a placeholder function. Implement categorization logic based
on the website's content.
    return 'Uncategorized'

def extract_info(url):
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        meta_title, meta_description = get_meta_info(soup)
        tech_stack = get_tech_stack(soup)
        social_media_links = get_social_media_links(soup)
        payment_gateways = get_payment_gateways(soup)
        website_language = get_website_language(soup)
        category = categorize_website(soup)

        # Insert data into MySQL
        cursor.execute("""
        INSERT INTO websites (url, meta_title, meta_description,
tech_stack, payment_gateways, website_language, category,
social_media_links)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
        """, (url, meta_title, meta_description, tech_stack,
payment_gateways, website_language, category, social_media_links))
        db.commit()
        print(f"Data for {url} inserted successfully.")
    except Exception as e:
        print(f"Error extracting info from {url}: {e}")

# Example list of websites
websites = [
    'https://example1.com',
    'https://example2.com',
    # Add more websites
]

for site in websites:
    extract_info(site)

cursor.close()
db.close()
```

Step 2: Data Storage

SQL Script to Create Tables

```
sql
Copy code
```

```
CREATE DATABASE IF NOT EXISTS web_info;

USE web_info;

CREATE TABLE IF NOT EXISTS websites (
    id INT AUTO_INCREMENT PRIMARY KEY,
    url VARCHAR(255),
    meta_title VARCHAR(255),
    meta_description TEXT,
    tech_stack TEXT,
    payment_gateways TEXT,
    website_language VARCHAR(50),
    category VARCHAR(50),
    social_media_links TEXT
);
```

Step 3: Documentation

README.md

~~~
markdown
Copy code
# Web Information Extractor

## Description

This project extracts information from a list of websites, including social
media links, tech stack, meta title, meta description, payment gateways,
website language, and category. The extracted information is stored in a
MySQL database.

## Setup

1. Install required Python libraries:
   ```bash
   pip install requests beautifulsoup4 selenium mysql-connector-python
~~~

2. Download and install the appropriate WebDriver for your browser (e.g., ChromeDriver for Chrome).
3. Set up MySQL database:
   - Run the provided SQL script to create the necessary tables.
4. Update the database connection details in the script (`user`, `password`, `host`, and `database`).
5. Run the script:

   ```
   bash
   Copy code
   python extract_info.py
   ```

Files

- `extract_info.py`: Python script to scrape website information and store it in a MySQL database.
- `create_tables.sql`: SQL script to create the necessary tables in the MySQL database.

- `README.md`: This documentation file.

Challenges

- Handling dynamic content with Selenium for some websites.
- Categorizing websites based on their content (placeholder function provided).

Notes

- Ensure the WebDriver is in your system's PATH.
- Customize the `categorize_website` function as needed for better categorization logic.

Author

[Your Name]

~~~
shell
Copy code

## Step 4: Code Quality

Ensure that your code is well-commented, follows PEP 8 standards, and is
modular. Use functions to separate different parts of the logic, making it
easier to maintain and understand.

## Submission

- Source code (`extract_info.py`)
- SQL script (`create_tables.sql`)
- README file (`README.md`)
- Brief report (can be included in README)

### Example Brief Report

```markdown
## Report

### Approach

1. **Web Scraping**: Used `BeautifulSoup` for parsing HTML and `Selenium`
for handling dynamic content.
2. **Data Extraction**: Extracted meta information, tech stack, social
media links, payment gateways, website language, and category.
3. **Data Storage**: Stored the extracted information in a MySQL database.

### Challenges

- **Dynamic Content**: Some websites use JavaScript to load content
dynamically. Handled using Selenium.
- **Categorization**: Categorizing websites based on content is complex and
requires a more sophisticated approach.

### Improvements
~~~

```
- Implement a more robust categorization algorithm.
- Add error handling and logging for better debugging and maintenance.

[Your Name]
```

This comprehensive approach ensures the extraction and storage of the required information efficiently and effectively.

Enhanced Web Scraping Code

Below is an improved version of the `extract_info.py` script with additional features:

1. **Error Handling**: Improved error handling to ensure the script continues running even if it encounters issues with some websites.
2. **Logging**: Added logging for better tracking of the script's execution.
3. **Dynamic Content Handling**: Use Selenium for websites that load content dynamically.

```
python
Copy code
import requests
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import mysql.connector
import re
import logging

# Setup logging
logging.basicConfig(filename='web_scraping.log', level=logging.INFO,
format='%(asctime)s %(message)s')

# Setup MySQL connection
db = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="web_info"
)
cursor = db.cursor()

# Create tables
cursor.execute("""
CREATE TABLE IF NOT EXISTS websites (
    id INT AUTO_INCREMENT PRIMARY KEY,
    url VARCHAR(255),
    meta_title VARCHAR(255),
    meta_description TEXT,
    tech_stack TEXT,
    payment_gateways TEXT,
    website_language VARCHAR(50),
    category VARCHAR(50),
    social_media_links TEXT
)
""")

def get_meta_info(soup):
    title = soup.find('title').text if soup.find('title') else ''
    description = ''
    if soup.find('meta', attrs={'name': 'description'}):
        description = soup.find('meta', attrs={'name':
'description'})['content']
    return title, description

def get_tech_stack(soup):
    tech_stack = set()
    scripts = soup.find_all('script')
    for script in scripts:
        src = script.get('src', '').lower()
        if 'react' in src:
            tech_stack.add('React')
        elif 'angular' in src:
            tech_stack.add('Angular')
        elif 'vue' in src:
            tech_stack.add('Vue.js')
        elif 'jquery' in src:
            tech_stack.add('jQuery')
    return ', '.join(tech_stack)

def get_social_media_links(soup):
    social_links = []
    for a in soup.find_all('a', href=True):
        href = a['href']
        if 'facebook.com' in href:
            social_links.append('Facebook: ' + href)
        elif 'twitter.com' in href:
            social_links.append('Twitter: ' + href)
        elif 'linkedin.com' in href:
            social_links.append('LinkedIn: ' + href)
        elif 'instagram.com' in href:
            social_links.append('Instagram: ' + href)
    return ', '.join(social_links)

def get_payment_gateways(soup):
    text = soup.text.lower()
    gateways = []
    if 'paypal' in text:
        gateways.append('PayPal')
    if 'stripe' in text:
        gateways.append('Stripe')
    if 'razorpay' in text:
        gateways.append('Razorpay')
    return ', '.join(gateways)

def get_website_language(soup):
    if soup.find('html'):
        return soup.find('html').get('lang', '')
    return ''

def categorize_website(soup):
    # Implementing a basic categorization logic based on keywords in the
meta tags
    meta_keywords = ''
    if soup.find('meta', attrs={'name': 'keywords'}):
        meta_keywords = soup.find('meta', attrs={'name':
'keywords'})['content'].lower()

    if 'e-commerce' in meta_keywords:
        return 'E-commerce'
    elif 'blog' in meta_keywords:
        return 'Blog'
    elif 'news' in meta_keywords:
        return 'News'
    elif 'portfolio' in meta_keywords:
        return 'Portfolio'
    else:
        return 'Uncategorized'

def extract_info(url):
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        # If the page is dynamically loaded, use Selenium
        if not soup.find('title'):
            service = Service(ChromeDriverManager().install())
            driver = webdriver.Chrome(service=service)
            driver.get(url)
            soup = BeautifulSoup(driver.page_source, 'html.parser')
            driver.quit()

        meta_title, meta_description = get_meta_info(soup)
        tech_stack = get_tech_stack(soup)
        social_media_links = get_social_media_links(soup)
        payment_gateways = get_payment_gateways(soup)
        website_language = get_website_language(soup)
        category = categorize_website(soup)

        # Insert data into MySQL
        cursor.execute("""
        INSERT INTO websites (url, meta_title, meta_description,
tech_stack, payment_gateways, website_language, category,
social_media_links)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
        """, (url, meta_title, meta_description, tech_stack,
payment_gateways, website_language, category, social_media_links))
        db.commit()
        logging.info(f"Data for {url} inserted successfully.")
    except Exception as e:
        logging.error(f"Error extracting info from {url}: {e}")

# Example list of websites
websites = [
    'https://example1.com',
    'https://example2.com',
    # Add more websites
]

for site in websites:
    extract_info(site)

cursor.close()
db.close()
```

Enhanced SQL Script

```
sql
Copy code
```

```
CREATE DATABASE IF NOT EXISTS web_info;

USE web_info;

CREATE TABLE IF NOT EXISTS websites (
    id INT AUTO_INCREMENT PRIMARY KEY,
    url VARCHAR(255),
    meta_title VARCHAR(255),
    meta_description TEXT,
    tech_stack TEXT,
    payment_gateways TEXT,
    website_language VARCHAR(50),
    category VARCHAR(50),
    social_media_links TEXT
);
```

Enhanced README.md

~~~
markdown
Copy code
# Web Information Extractor

## Description

This project extracts information from a list of websites, including social
media links, tech stack, meta title, meta description, payment gateways,
website language, and category. The extracted information is stored in a
MySQL database.

## Setup

1. **Install required Python libraries:**
   ```bash
   pip install requests beautifulsoup4 selenium mysql-connector-python
webdriver-manager
~~~

2. **Download and install the appropriate WebDriver for your browser (e.g., ChromeDriver for Chrome).**
    - Ensure the WebDriver is in your system's PATH.
    - For Chrome, you can use `webdriver-manager` which is included in the script.
3. **Set up MySQL database:**
    - Run the provided SQL script to create the necessary tables.

```
sql
Copy code
CREATE DATABASE IF NOT EXISTS web_info;

USE web_info;

CREATE TABLE IF NOT EXISTS websites (
    id INT AUTO_INCREMENT PRIMARY KEY,
    url VARCHAR(255),
    meta_title VARCHAR(255),
    meta_description TEXT,
    tech_stack TEXT,
    payment_gateways TEXT,
    website_language VARCHAR(50),
    category VARCHAR(50),
```

```
        social_media_links TEXT
    );
```

4. **Update the database connection details in the script (`user`, `password`, `host`, and `database`).**
5. **Run the script:**

   ```
   bash
   Copy code
   python extract_info.py
   ```

Files

- `extract_info.py`: Python script to scrape website information and store it in a MySQL database.
- `create_tables.sql`: SQL script to create the necessary tables in the MySQL database.
- `README.md`: This documentation file.

Challenges

- Handling dynamic content with Selenium for some websites.
- Categorizing websites based on their content (a basic placeholder function is provided).

Improvements

- Implement a more robust categorization algorithm based on content analysis.
- Add more sophisticated error handling and logging for better debugging and maintenance.

Notes

- Ensure the WebDriver is in your system's PATH.
- Customize the `categorize_website` function as needed for better categorization logic.

Author

[Your Name]

~~~
csharp
Copy code

### Enhanced Report

```markdown
## Report

### Approach
~~~

```
1. **Web Scraping**: Used `BeautifulSoup` for parsing HTML and `Selenium`
for handling dynamic content.
2. **Data Extraction**: Extracted meta information, tech stack, social
media links, payment gateways, website language, and category.
3. **Data Storage**: Stored the extracted information in a MySQL database.

### Challenges

- **Dynamic Content**: Some websites use JavaScript to load content
dynamically. This was handled using Selenium.
- **Categorization**: Categorizing websites based on content is complex. A
basic categorization based on meta keywords was implemented as a
placeholder.

### Improvements

- Implement a more robust categorization algorithm based on deeper content
analysis.
- Enhance error handling and logging to capture more detailed information
for debugging and maintenance.

### Lessons Learned

- Combining `BeautifulSoup` and `Selenium` provides a powerful solution for
web scraping both static and dynamic content.
- Logging is crucial for tracking the execution and identifying issues
during the scraping process.

## Notes

- Ensure the WebDriver is correctly installed and in your system's PATH.
- Adjust the `categorize_website` function to fit specific needs for better
categorization accuracy.

[Your Name]
```

This comprehensive approach ensures the extraction, categorization, and storage of the required information in an efficient and effective manner. The enhanced logging and error handling improve the robustness of the solution, and detailed documentation provides clear instructions for setup and usage.