Tạo Web tìm kiếm thông tin

Phần 1 : Xây dựng Database Mysql liên quan đến các tin tức trong ngày. Crawl tin tức hàng ngày từ các website được cho sẵn

https://dantri.com.vn/rss.htm
https://vnexpress.net/rss/tin-moi-nhat.rss
https://cafebiz.vn/index.rss
https://tinhte.vn/tag/tinhtevn-rss
https://www.24h.com.vn/guest/RSS/

Mục Tiêu : Thực hành với thư viện Request trong Python để tạo trang Web, Biết lấy giữ liệu từ file RSS vào Database của mình

1. Thư viện Request trong Python

1. **Requests** là một thư viện Python cho phép bạn gửi các yêu cầu HTTP (GET, POST, PUT, DELETE, v.v.) một cách dễ dàng. Thư viện này được thiết kế để đơn giản hóa việc gửi yêu cầu HTTP và xử lý các phản hồi từ server.
2. **Công dụng** :

- **Gửi yêu cầu HTTP**: Bạn có thể gửi các loại yêu cầu như GET, POST, PUT, DELETE, v.v.
- **Xử lý phản hồi**: Thư viện giúp bạn dễ dàng đọc và xử lý phản hồi từ server, bao gồm nội dung, mã trạng thái, tiêu đề, v.v.
- **Hỗ trợ xác thực**: Requests hỗ trợ nhiều phương thức xác thực như Basic Auth, OAuth, v.v.
- **Quản lý Session**: Bạn có thể quản lý các phiên làm việc (session) để lưu trữ thông tin giữa các yêu cầu.
- **Gửi dữ liệu**: Hỗ trợ gửi dữ liệu qua form hoặc dưới dạng JSON

3. Để sử dụng thư viện Request chúng ta sẽ phải cài đặt thư viện

```
pip install requests
```

Cách sử dụng Request để lập trình ra một website tìm kiếm.

1. Chúng ta phải import các thư viên cần thiết để thực hiện chương trình
2. Gửi yêu cầu bằng các phương thức như GET, PORT, ... đến sever
3. Chạy chương trinh và ịn ra kết quả

import thư viện

```
from flask import Flask, request, jsonify
```

2. Lấy giữ liệu từ file RSS vào Database của mình

RSS (Really Simple Syndication hoặc Rich Site Summary) là một định dạng file được sử dụng để cung cấp nội dung cập nhật một cách nhanh chóng và dễ dàng từ các website, như tin tức, blog, podcast, và nhiều nguồn thông tin khác. File RSS chứa thông tin tóm tắt về nội dung mới nhất từ trang web, cho phép người dùng theo dõi mà không cần truy cập trực tiếp vào từng trang.

File RSS thường có định dạng XML, và bao gồm các thẻ chứa thông tin và các thẻ này nằm trong một cái `<item> </item>`

- **Title**: Tiêu đề của bài viết hoặc nội dung.
- **Link**: Liên kết đến bài viết hoặc nội dung đầy đủ trên trang web.
- **Description**: Mô tả ngắn gọn về nội dung.
- **PubDate**: Ngày xuất bản của bài viết.
- **Author**: Tác giả của bài viết (tuỳ chọn).

Để insert thông tin từ file RSS vào Database chúng ta phải tạo bảng chứa các trường thông tin trong các file RSS mà bạn được cung cấp ví dụ như tôi tạo bảng **dantri_news** thì các thường của nó phải theo file RSS

```
CREATE TABLE newsdb.dantri_news (
 id int NOT NULL AUTO_INCREMENT,
 title varchar(255) NOT NULL,
 pub_date datetime DEFAULT NULL,
 link varchar(255) NOT NULL,
 guid varchar(255) NOT NULL,
 description text DEFAULT NULL,
 creator varchar(255) DEFAULT NULL,
 category varchar(255) DEFAULT NULL,
 PRIMARY KEY (id)
)
```

Các thư viện cần để insert thông tin từ file RSS vào Database

```
import requests
import mysql.connector
from bs4 import BeautifulSoup
from datetime import datetime
import html
```

Chúng ta phải tạo các bảng trong MySQL trước khi thực hiện chương trình

Đầu tiên chúng ta hãy tạo một hàm để nhận vào một URL của feed RSS

- Gửi yêu cầu GET để lấy nội dung của feed.
- Sử dụng BeautifulSoup để phân tích cú pháp nội dung XML.
- Lặp qua tất cả các phần tử `<item>` và lấy các thông tin cần thiết như tiêu đề, liên kết, mô tả, ngày xuất bản, và các trường bổ sung khác.
- Tùy thuộc vào nguồn RSS, thông tin sẽ được thêm vào danh sách `news_list` theo các định dạng khác nhau.
- Nếu có lỗi xảy ra trong quá trình phân tích, hàm sẽ in ra thông báo lỗi và trả về danh sách rỗng.

```
def parse_rss(url):
    try:
        response = requests.get(url)
        response.encoding = 'utf-8'
        clean_content = html.unescape(response.text)
        soup = BeautifulSoup(clean_content, "xml")

        news_list = []
        for item in soup.find_all("item"):
            # Lấy thông tin chung
            title = item.title.text if item.title else ''
            link = item.link.text if item.link else ''
            description = item.description.text if item.description else ''
            pub_date = item.pubDate.text if item.pubDate else None
            guid = item.guid.text if item.guid else ''

            # Xử lý pub_date
            if pub_date:
                pub_date = pub_date.split(' ')[0:5]
                pub_date = ' '.join(pub_date)
                try:
                    pub_date = datetime.strptime(pub_date, "%a, %d %b %Y %H:%M:%S")
                except ValueError:
                    pub_date = None

            print(f"Parsing item: {title}, link: {link}, pub_date: {pub_date}")

            # Kiểm tra và lấy giá trị cho từng trường
            author = item.find("author").text if item.find("author") else None
            content_encoded = item.find("{http://purl.org/rss/1.0/modules/content/}encoded").text if item.find("
{http://purl.org/rss/1.0/modules/content/}encoded") else None
            comments_count = int(item.find("{http://purl.org/rss/1.0/modules/slash/}comments").text) if item.find("
{http://purl.org/rss/1.0/modules/slash/}comments") else 0
```

```
            image = item.find("image").text if item.find("image") else None
            enclosure = item.find("enclosure")
            enclosure_url = enclosure["url"] if enclosure else None
            enclosure_type = enclosure["type"] if enclosure else None
            enclosure_length = int(enclosure["length"]) if enclosure and "length" in enclosure.attrs else None

            # Thêm vào news_list tùy thuộc vào nguồn RSS
            if "dantri.com.vn" in url:
                creator = item.find("{http://purl.org/dc/elements/1.1/}creator").text if item.find("{http://purl.org/dc/elements/1.1/}creator") else None
                category = item.find("category").text if item.find("category") else None
                news_list.append((title, pub_date, link, guid, description, creator, category))
            elif "vnexpress.net" in url:
                news_list.append((title, description, pub_date, link, guid, enclosure_url, enclosure_type, enclosure_length))
            elif "cafebiz.vn" in url:
                news_list.append((title, description, pub_date, link, guid))
            elif "tinhte.vn" in url:
                news_list.append((title, description, pub_date, link, guid, author, content_encoded, comments_count, image))
            elif "24h.com.vn" in url:
                news_list.append((title, description, link, guid, pub_date))

        print(f"Found {len(news_list)} items in {url}")
        return news_list

    except Exception as e:
        print(f"Lỗi khi parse XML từ URL {url}: {e}")
        return []
```

Insert vào Database

```
def insert_news(news_list, table_name):
    # Tạo câu lệnh SQL tương ứng với bảng
    if table_name == "dantri_news":
        sql = f"""
```

```
        INSERT INTO dantri_news (title, pub_date, link, guid, description, creator, category)        VALUES (%s, %s, %s, %s, %s, %s, %s)        """
    elif table_name == "vnexpress_news":
        sql = f"""
        INSERT INTO {table_name} (title, description, pub_date, link, guid, enclosure_url, enclosure_type, enclosure_length)        VALUES (%s, %s, %s, %s,
%s, %s, %s, %s)        """
    elif table_name == "cafebiz_news":
        sql = f"""
        INSERT INTO {table_name} (title, description, pub_date, link, guid)        VALUES (%s, %s, %s, %s, %s)        """
    elif table_name == "tinhte_news":
        sql = f"""
        INSERT INTO {table_name} (title, description, pub_date, link, guid, author, content_encoded, comments_count, image)        VALUES (%s, %s, %s,
%s, %s, %s, %s, %s, %s)        """
    elif table_name == "twentyfourh_news":
        sql = f"""
        INSERT INTO {table_name} (title, description, link, guid, pub_date)        VALUES (%s, %s, %s, %s, %s)        """

    cursor.executemany(sql, news_list)
    db.commit()

# Lặp qua các RSS và lưu vào DB
for url, table_name in rss_data.items():
    news_data = parse_rss(url)
    if news_data:
        insert_news(news_data, table_name)
# Đóng kết nối
cursor.close()
db.close()
```

Phần 2 :

Yêu cầu:

1. Xây dựng API dạng GET tìm kiếm tin tức theo keyword (có thể sáng tạo việc chọn keyword) và trả về kết quả.

2. Lưu keyword được tìm kiếm về 1 bảng dữ liệu.

3.Viết một trang .html có những tính năng sau:

- Có chức năng tìm kiếm sử dụng API ở trên dùng javascript

- Hiển thị kết quả tìm kiếm cho người dùng.

- Hiển thị các từ khóa hay được tìm kiếm nhiều nhất (gọi API để lấy thông tin trong database).

- Hiển thị các tin mới nhất (gọi API trong db).

Mục Tiêu : Làm Quen với HTML và Javascript, Tạo ra các API với các tác dụng khác nhau và kết nối các API với HTML

1. Xây dựng các API

Để xây dựng các API chúng ta phải thực hiện các bước như sau :

1. Gửi yêu cầu Request gồm địa chỉ URL thông qua HTTP.
2. Yêu cầu Request chúng ta có thể sử dụng phương thức GET,PORT,...
3. **API** nhận yêu cầu và thực hiện các thao tác cần thiết, sau đó trả về phản hồi (response) cho client.

Phần tạo các API theo yêu cầu của bài

- Import thư viện

```
from flask import Flask, request, jsonify
import mysql.connector
from flask_cors import CORS
```

- Gửi phương thức GET để tìm từ khóa trong bảng

```
app = Flask(__name__)
CORS(app)
@app.route('/search_news', methods=['GET'])
def search_news():
    keyword = request.args.get('keyword')
    if not keyword:
        return jsonify({"error": "Keyword is required"}), 400

    # Danh sách các bảng và cột cần tìm kiếm
    tables = [
        {"name": "dantri_news", "columns": ["title", "description"]},
        {"name": "vnexpress_news", "columns": ["title", "description"]},
        {"name": "cafebiz_news", "columns": ["title", "description"]},
        {"name": "twentyfourh_news", "columns": ["title", "description"]},
        {"name": "tinhte_news", "columns": ["title", "description"]},
    ]
    results = []

    # Duyệt qua từng bảng và thực hiện tìm kiếm
    for table in tables:
        table_name = table["name"]
        columns = table["columns"]

        # Tạo câu truy vấn SQL tìm kiếm với điều kiện LIKE cho cả title và description
        query = f"SELECT * FROM {table_name} WHERE {columns[0]} LIKE %s OR {columns[1]} LIKE %s"
        cursor = db.cursor(dictionary=True)
```

```
        try:
            # Thực thi truy vấn với từ khóa tìm kiếm
            cursor.execute(query, (f"%{keyword}%", f"%{keyword}%"))
            records = cursor.fetchall()
            results.extend(records)
        finally:
            close_cursor(cursor)

    # Lưu từ khóa tìm kiếm vào bảng `search_keywords`
    save_keyword_to_db(keyword)

    return jsonify({"results": results})
```

- Thực hiện phương thức PORT để lưu những từ khóa vừa tìm lại

```
@app.route('/save_keyword', methods=['POST'])
def save_keyword():
    data = request.json
    keyword = data.get('keyword')
    if keyword:
        save_keyword_to_db(keyword)
        return jsonify({"message": "Keyword saved successfully"})
    else:
        return jsonify({"error": "Keyword is required"}), 400


# Hàm lưu từ khóa vào bảng `search_keywords`
def save_keyword_to_db(keyword):
    cursor = db.cursor()
    try:
        query = "INSERT INTO search_keywords (keyword) VALUES (%s)"
        cursor.execute(query, (keyword,))
```

```
        db.commit()
    finally:
        close_cursor(cursor)
```

- Lấy các từ khóa tìm kiếm gần đây, sử dụng phương thức GET

```
@app.route('/get_recent_keywords', methods=['GET'])
def get_recent_keywords():
    cursor = db.cursor(dictionary=True)
    try:
        query = "SELECT keyword FROM search_keywords ORDER BY id DESC LIMIT 10"
        cursor.execute(query)
        keywords = cursor.fetchall()
    finally:
        close_cursor(cursor)

    return jsonify({"keywords": [kw['keyword'] for kw in keywords]})
```

- Lấy ra các từ khóa hay tìm kiếm nhất, sử dụng phương thức GET

```
@app.route('/top_keywords_news', methods=['GET'])
def top_keywords_news():
    cursor = db.cursor(dictionary=True)

    # Lấy top 3 từ khóa được tìm kiếm nhiều nhất từ bảng `search_keywords`
    query_top_keywords = """
        SELECT keyword, COUNT(keyword) as count        FROM search_keywords
        GROUP BY keyword
        ORDER BY count DESC
        LIMIT 3
    """
    cursor.execute(query_top_keywords)
```

```
top_keywords = [row['keyword'] for row in cursor.fetchall()]

if not top_keywords:
    return jsonify({"message": "Không tìm thấy từ khóa nào"}), 404

tables = [
    {"name": "dantri_news", "columns": ["title", "description"]},
    {"name": "vnexpress_news", "columns": ["title", "description"]},
    {"name": "cafebiz_news", "columns": ["title", "description"]},
    {"name": "twentyfourh_news", "columns": ["title", "description"]},
    {"name": "tinhte_news", "columns": ["title", "description"]},
]
results = []

# Tìm kiếm một tin tức duy nhất liên quan đến mỗi từ khóa trong top 3
for keyword in top_keywords:
    keyword_results = {"keyword": keyword, "news": None}
    for table in tables:
        table_name = table["name"]
        columns = table["columns"]
        # Câu truy vấn tìm kiếm tin tức theo từ khóa, giới hạn chỉ lấy 1 kết quả
        query_news = f"""
            SELECT * FROM {table_name} WHERE {columns[0]} LIKE %s OR {columns[1]} LIKE %s
            LIMIT 1
        """            cursor.execute(query_news, (f"%{keyword}%", f"%{keyword}%"))
        record = cursor.fetchone()
        if record:
            keyword_results["news"] = record
            break  # Nếu đã tìm thấy tin tức cho từ khóa này, thoát khỏi vòng lặp

    # Thêm kết quả tìm được vào `results`        results.append(keyword_results)

close_cursor(cursor)
return jsonify({"top_keywords_news": results})
```

2. HTML và Javascript

2.1 HTLM

HTML (HyperText Markup Language) là ngôn ngữ đánh dấu chuẩn được sử dụng để tạo và thiết kế các trang web. HTML định nghĩa cấu trúc của một trang web bằng cách sử dụng các thẻ (tags) để xác định các phần tử khác nhau trên trang, như tiêu đề, đoạn văn, hình ảnh, liên kết, và nhiều thành phần khác.

Các thẻ là các thành phần cơ bản của HTML, được bao bọc trong dấu ngoặc nhọn (`< >`). Ví dụ:

- `<h1>` : Thẻ tiêu đề cấp 1.
- `<p>` : Thẻ đoạn văn.
- `<img>` : Thẻ hình ảnh

Đây là khung của HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

</body>
</html>
```

Các thẻ ở phần đầu của HTML, cung cấp thông tin về trang web, định nghĩa cách nó sẽ được hiển thị và tương tác trên các thiết bị khác nhau.

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Tìm Kiếm Tin Tức</title>
<link rel="stylesheet" href="styles.css">
```

CSS HTML là một ngôn ngữ định dạng được sử dụng để thiết kế và bố trí các trang web. CSS cho phép bạn điều chỉnh cách hiển thị các phần tử HTML bằng cách xác định các thuộc tính như màu sắc, kích thước, kiểu chữ, khoảng cách, và nhiều yếu tố khác.

Khối CSS trong được sử dụng trong bài để điều chỉnh cho dễ nhìn

```
<style>
    body {
        font-family: Arial, sans-serif;
        margin: 20px;
    }

    .search-container {
        display: flex;
        align-items: center;
        margin-bottom: 20px;
    }

    #search-box {
        flex: 1;
        padding: 10px;
        border: 1px solid #ccc;
        border-radius: 4px;
        margin-right: 10px;
    }

    #search-button {
        padding: 10px 15px;
```

```
    background-color: #007BFF;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

#search-button:hover {
    background-color: #0056b3;
}

.suggestions {
    border: 1px solid #ccc;
    border-radius: 4px;
    max-height: 150px;
    overflow-y: auto;
    position: absolute;
    background-color: white;
    width: calc(100% - 20px);
    z-index: 10;
    margin-top: 5px;
}

.suggestion-item {
    padding: 8px;
    cursor: pointer;
}

.suggestion-item:hover {
    background-color: #f0f0f0;
}

.result-item {
    border: 1px solid #ccc;
```

```
        border-radius: 4px;
        margin: 10px 0;
        padding: 10px;
    }
</style>
```

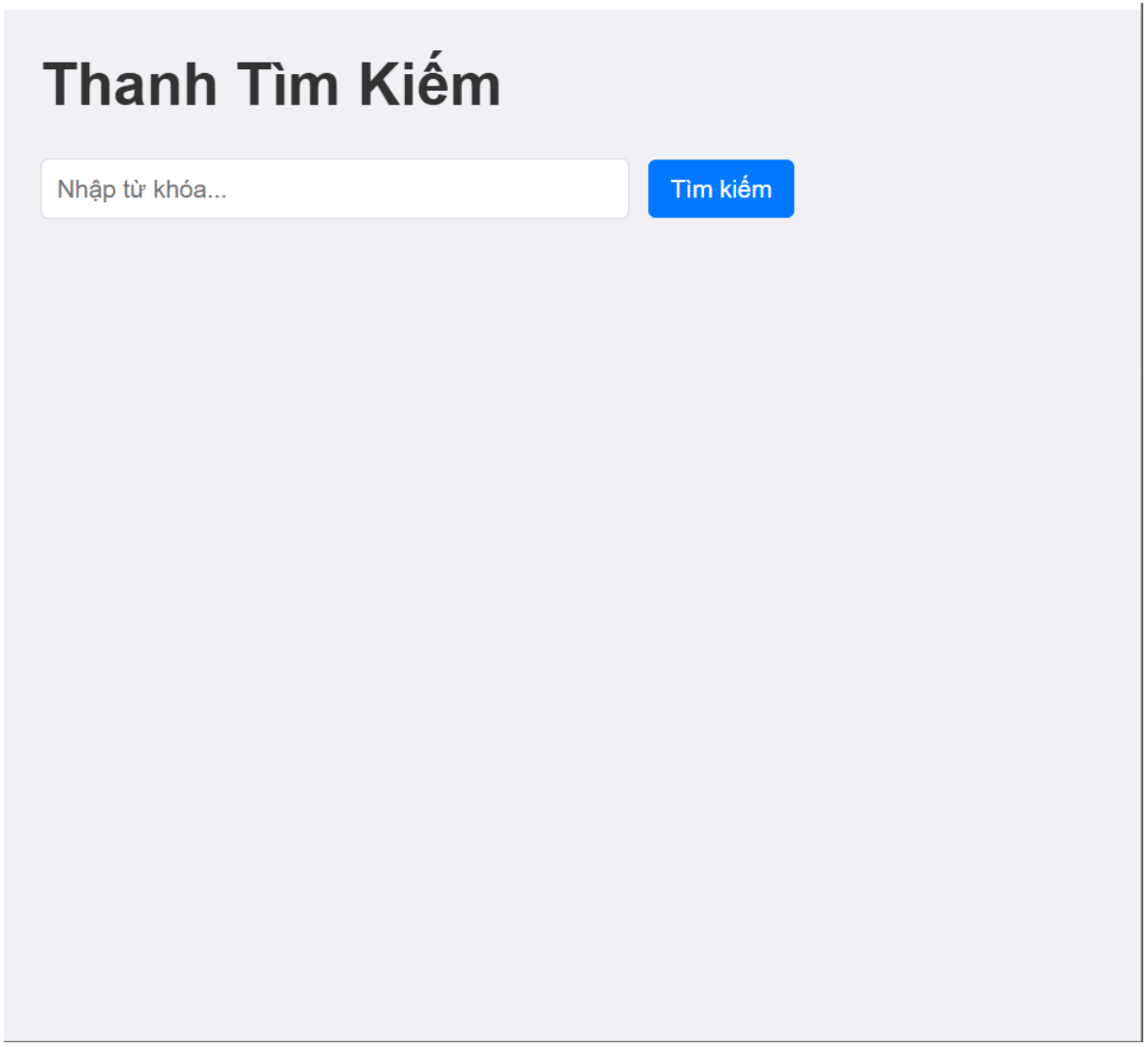
Như yêu cầu của đề bài là tạo một trang HTML có chức năng tìm kiếm thì đây chính là tất cả các thẻ tạo thành phần nội dung chính của một trang web tìm kiếm tin tức.

```
<h1>Tìm Kiếm Tin Tức</h1>

<div class="search-container">
    <input type="text" id="search-box" placeholder="Nhập từ khóa tìm kiếm..." />
    <button id="search-button">Tìm kiếm</button>
</div>

<div id="suggestions" class="suggestions"></div>
<div id="results-list"></div>

<h2>Top Tìm Kiếm Nhiều Nhất</h2>
<ul id="results"></ul>
```

Đây là phần kết nối với Javascript

```
<script src="hienthikeyword.js"></script>
```

Khi thực hiện chạy chương trình, sẽ xuất hiện một giao diện tìm kiếm bằng từ khóa và sẽ hiện ra kết quả ở dưới.



2.2 Javascript

Javascript : là một ngôn ngữ lập trình được thiết kế để thêm tính tương tác cho các trang web.
JavaScript : cho phép bạn viết mã mà không cần chỉ định kiểu dữ liệu cụ thể cho các biến, giúp lập trình trở nên linh hoạt hơn.

HTML và Javascript có liên kết chặt chẽ với nhau thông qua việc nhúng thẻ `<script></script>`. Điều này có thể được thực hiện ở trong `<head>` hoặc ở cuối `<body>` của tài liệu HTML.
Chúng ta có thể nhúng trực tiếp đoạn code Javascript vào HTML hoặc nhúng gián tiếp qua file của Javascrip

Theo yêu cầu của bài sử dụng việc nhúng qua file để không bị rối và dễ sử lí lỗi

```
<script src="hienthikeyword.js"></script>
```

DOM, viết tắt của **Document Object Model**, là một mô hình lập trình mà trình duyệt sử dụng để đại diện cho cấu trúc của tài liệu HTML hoặc XML. Nó cho phép lập trình viên truy cập và thao tác với nội dung, cấu trúc và kiểu dáng của tài liệu theo cách linh hoạt.
Đây là đoạn code lấy các phần tử từ DOM trong yêu cầu của bài tập ở trên

```
const searchButton = document.getElementById("search-button");
const searchBox = document.getElementById("search-box");
const suggestionsContainer = document.getElementById("suggestions");
const resultsList = document.getElementById("results-list");
```

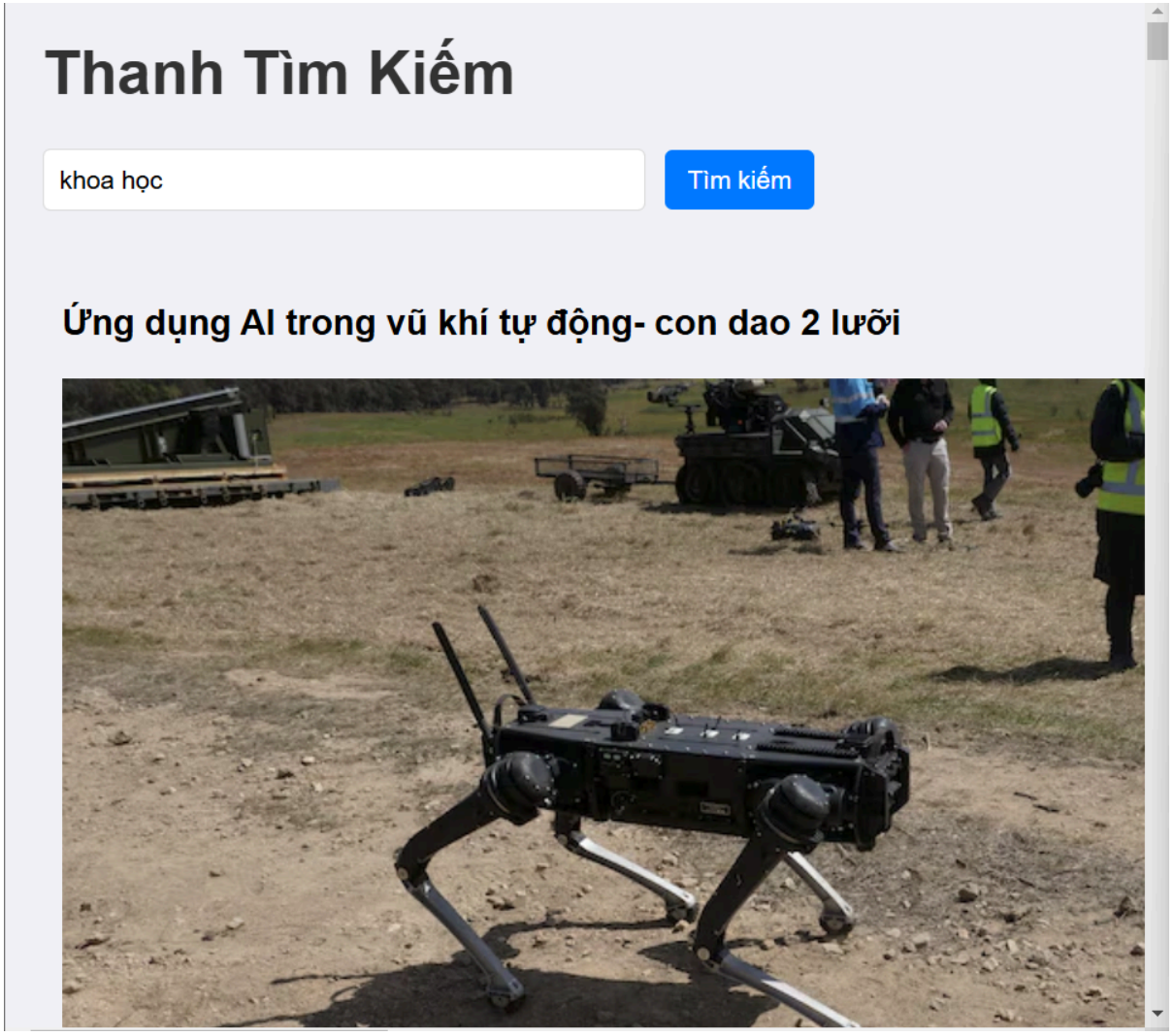
Hàm tìm kiếm tin tức trong Javascript
Khi hàm này được thực hiện thì khi chúng ta nhập vào từ khóa nào đó thì lập tức nó sẽ thực hiện tìm kiếm từ khó đó có trong Database của chúng ta không và hiển thị kết quả ra HTML

```
async function searchNews() {
    const keyword = searchBox.value.trim();
    resultsList.innerHTML = ''; // Xóa kết quả cũ
    suggestionsContainer.innerHTML = ''; // Ẩn gợi ý tìm kiếm

    if (keyword === '') {
        resultsList.innerHTML = '<p>Vui lòng nhập từ khóa tìm kiếm.</p>';
        return;
    }
```

```
    try {
        const response = await fetch(`http://127.0.0.1:5000/search_news?keyword=${encodeURIComponent(keyword)}`);

        if (!response.ok) {
            throw new Error("Không tìm thấy tin tức hoặc có lỗi xảy ra");
        }

        const data = await response.json();

        if (data.results && data.results.length === 0) {
            resultsList.innerHTML = '<p>Không tìm thấy kết quả nào.</p>';
            return;
        }

        // Hiển thị kết quả
        data.results.forEach(item => {
            const resultItem = document.createElement('div');
            resultItem.classList.add('result-item');
            resultItem.innerHTML = `
                <h3>${item.title}</h3>
                <p>${item.description}</p>
                <a href="${item.link}" target="_blank">Xem chi tiết</a>
                <p><small>Ngày xuất bản: ${item.pub_date}</small></p>
            `;
            resultsList.appendChild(resultItem);
        });

    } catch (error) {
        resultsList.innerHTML = `<p>${error.message}</p>`;
        console.error('Lỗi:', error);
    }
}
```

```
// Thêm sự kiện cho nút tìm kiếm
searchButton.addEventListener("click", searchNews);
```

Hình ảnh cho thấy màn hình đã hiển thị ra kết quả khi người dùng thực hiện tìm kiếm bằng từ khóa "khoa học"



Hàm Lấy từ khóa tìm kiếm gần đây

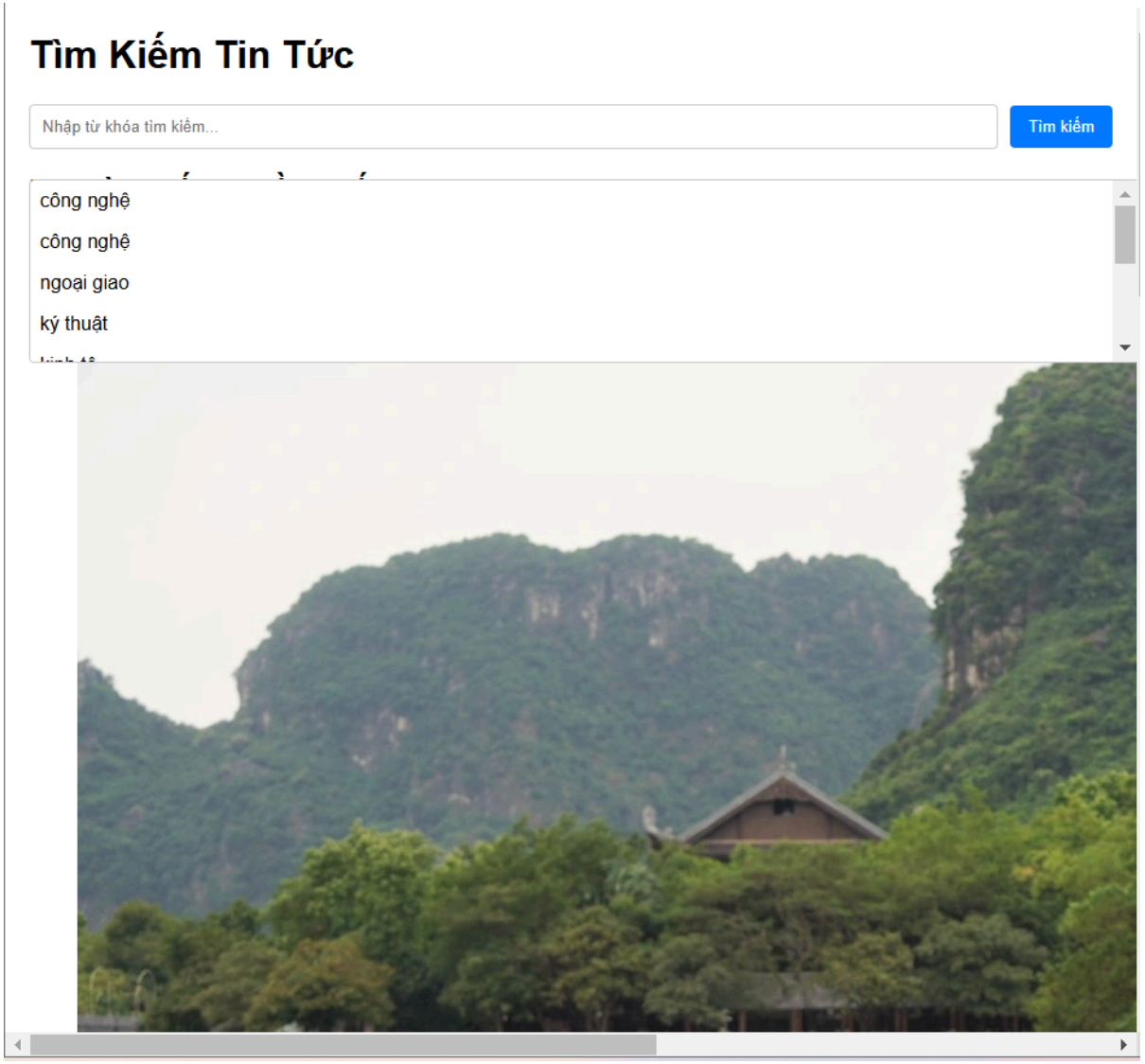
```
async function fetchRecentKeywords() {
    try {
        const response = await fetch('http://127.0.0.1:5000/get_recent_keywords');
        const data = await response.json();

        if (data.keywords) {
            data.keywords.forEach(keyword => {
                const suggestionItem = document.createElement('div');
                suggestionItem.classList.add('suggestion-item');
                suggestionItem.textContent = keyword;
                suggestionItem.addEventListener('click', () => {
                    searchBox.value = keyword; // Điền từ khóa vào ô tìm kiếm
                    searchNews(); // Thực hiện tìm kiếm
                });
                suggestionsContainer.appendChild(suggestionItem);
            });
        }
    } catch (error) {
        console.error('Lỗi khi lấy từ khóa gần đây:', error);
    }
}
```

Hàm để hiện gợi ý tìm kiếm khi nhập từ khóa

```
searchBox.addEventListener("input", () => {
    const keyword = searchBox.value.trim();
    if (keyword !== '') {
        suggestionsContainer.innerHTML = ''; // Xóa gợi ý cũ
        fetchRecentKeywords(); // Lấy gợi ý từ khóa gần đây
    } else {
        suggestionsContainer.innerHTML = ''; // Ẩn gợi ý khi không có từ khóa
```

```
    }
});
```

Hình ảnh này cho thấy được những từ khóa mà người dùng đã tìm gần đây



Hàm hiển thị top những tin nổi bật

```
async function fetchTopKeywords() {
    try {
        const response = await fetch('http://127.0.0.1:5000/top_keywords_news');
        if (!response.ok) {
            throw new Error('Không tìm thấy từ khóa nào');
        }
        const data = await response.json();
        displayResults(data.top_keywords_news);
    } catch (error) {
        console.error('Lỗi:', error);
        document.getElementById('results').innerHTML = `<p>${error.message}</p>`;
    }
}

function displayResults(keywords) {
    const resultsDiv = document.getElementById('results');
    resultsDiv.innerHTML = ''; // Xóa nội dung trước đó

    keywords.forEach(item => {
        const keywordItem = document.createElement('li');
        keywordItem.innerHTML = `<strong>Từ Khóa: ${item.keyword}</strong><br>
                                 ${item.news ? `<strong>Tin Tức:</strong><br>
                                 Tiêu đề: ${item.news.title}<br>
                                 Mô tả: ${item.news.description}` :
                                 'Không tìm thấy tin tức cho từ khóa này.'}`;
        resultsDiv.appendChild(keywordItem);
    });
}

window.onload = fetchTopKeywords; // Gọi hàm khi trang được tải
```

Đây là top những tin tức nổi bật mà người dùng hay tìm bằng từ khóa

Top Tìm Kiếm Nhiều Nhất

- **Từ Khóa: kinh tế**
  Tin Tức:
  Tiêu đề: Lượng hóa giá trị kinh tế di sản thế giới Tràng An