**IR ASSIGNMENT – 2**

**Build a Python Program To Create a Meta Crawler that Aggregates Search Results from Multiple Search Engines into A Unified List**

**Source Code :**

**Index.html:**

<!DOCTYPE html>

<html>

<head>

    <title>Meta Crawler</title>

    <style>

       input:focus{

        outline: 2px solid rgb(47, 255, 47);

       }

       .btn:hover{

        width: 40%;

        background-color: rgb(47, 255, 47);

        color: white;

        border: 3px solid   rgb(47, 255, 47);

       }

    </style>

</head>

<body style="background-color:#0D1B2A;">

    <h1 style="display:flex;justify-content: center; color:white;font-family: Arial, Helvetica, sans-serif;">Meta Search</h1>

    <form method="POST">

     <div style="display:flex;justify-content: center;">

        <input style="width: 50%; padding:12px ;border: none;border-radius: 7px;" type="text" name="search\_query" placeholder="Enter your search query">

        <button class="btn" style="margin-left:20px;background-color: rgb(47, 255, 47);color:white ;font-size: 15px; font-weight:700; padding:12px;border-radius: 9px;border: 3px solid rgb(47, 255, 47); width: 10%;" type="submit">Search</button>

     </div>

          </form>

    {% if results %}

    <h2 style="color: white;font-family: Arial, Helvetica, sans-serif;">Search Results:</h2>

    <ul>

        {% for result in results %}

        <li style="color: white;">

            <a style="color: white;" href="{{ result.link }}" target="\_blank">{{ result.title }}</a>

        </li>

        {% endfor %}

    </ul>

    {% endif %}

</body>

</html>

**App.py:**

import asyncio

import aiohttp

from urllib.parse import quote\_plus

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

GOOGLE\_API\_KEY = "AIzaSyAEltuEUs8rHL1IilDeeEjMtre6WY5Rqm0"

GOOGLE\_CX = "f2024a058c9ba4865"

BING\_API\_KEY = "YOUR\_BING\_API\_KEY"

async def fetch(session, url, headers=None):

    try:

        async with session.get(url, headers=headers) as response:

            return await response.json()

    except Exception as e:

        print(f"Error fetching {url}: {e}")

        return None

async def search\_google(query, num\_results=30):

    results = []

    for start in range(1, num\_results, 10):

        url = f"https://www.googleapis.com/customsearch/v1?q={quote\_plus(query)}&key={GOOGLE\_API\_KEY}&cx={GOOGLE\_CX}&num=10&start={start}"

        async with aiohttp.ClientSession() as session:

            data = await fetch(session, url)

            if data and "items" in data:

                results.extend([{"title": item["title"], "link": item["link"]} for item in data["items"]])

    return results[:num\_results]

async def search\_bing(query, num\_results=30):

    url = f"https://api.bing.microsoft.com/v7.0/search?q={quote\_plus(query)}&count={num\_results}"

    headers = {"Ocp-Apim-Subscription-Key": BING\_API\_KEY}

    async with aiohttp.ClientSession() as session:

        data = await fetch(session, url, headers)

        if data and "webPages" in data:

            return [{"title": item["name"], "link": item["url"]} for item in data["webPages"]["value"]]

    return []

async def search\_duckduckgo(query, num\_results=20):

    url = f"https://api.duckduckgo.com/?q={quote\_plus(query)}&format=json"

    async with aiohttp.ClientSession() as session:

        data = await fetch(session, url)

        if data and "RelatedTopics" in data:

            results = [{"title": item["Text"], "link": item["FirstURL"]} for item in data if "Text" in item and "FirstURL" in item]

            return results[:num\_results]

    return []

async def meta\_search(query, num\_results=60):

    google\_results, bing\_results, duckduckgo\_results = await asyncio.gather(

        search\_google(query, num\_results=30),

        search\_bing(query, num\_results=30),

        search\_duckduckgo(query, num\_results=20),

    )

    all\_results = google\_results + bing\_results + duckduckgo\_results

    unique\_results = []

    seen\_links = set()

    for result in all\_results:

        if result["link"] not in seen\_links:

            unique\_results.append(result)

            seen\_links.add(result["link"])

    return unique\_results[:num\_results]

@app.route("/", methods=["GET", "POST"])

async def index():

    results = []

    if request.method == "POST":

        search\_query = request.form.get("search\_query", "").strip()

        if search\_query:

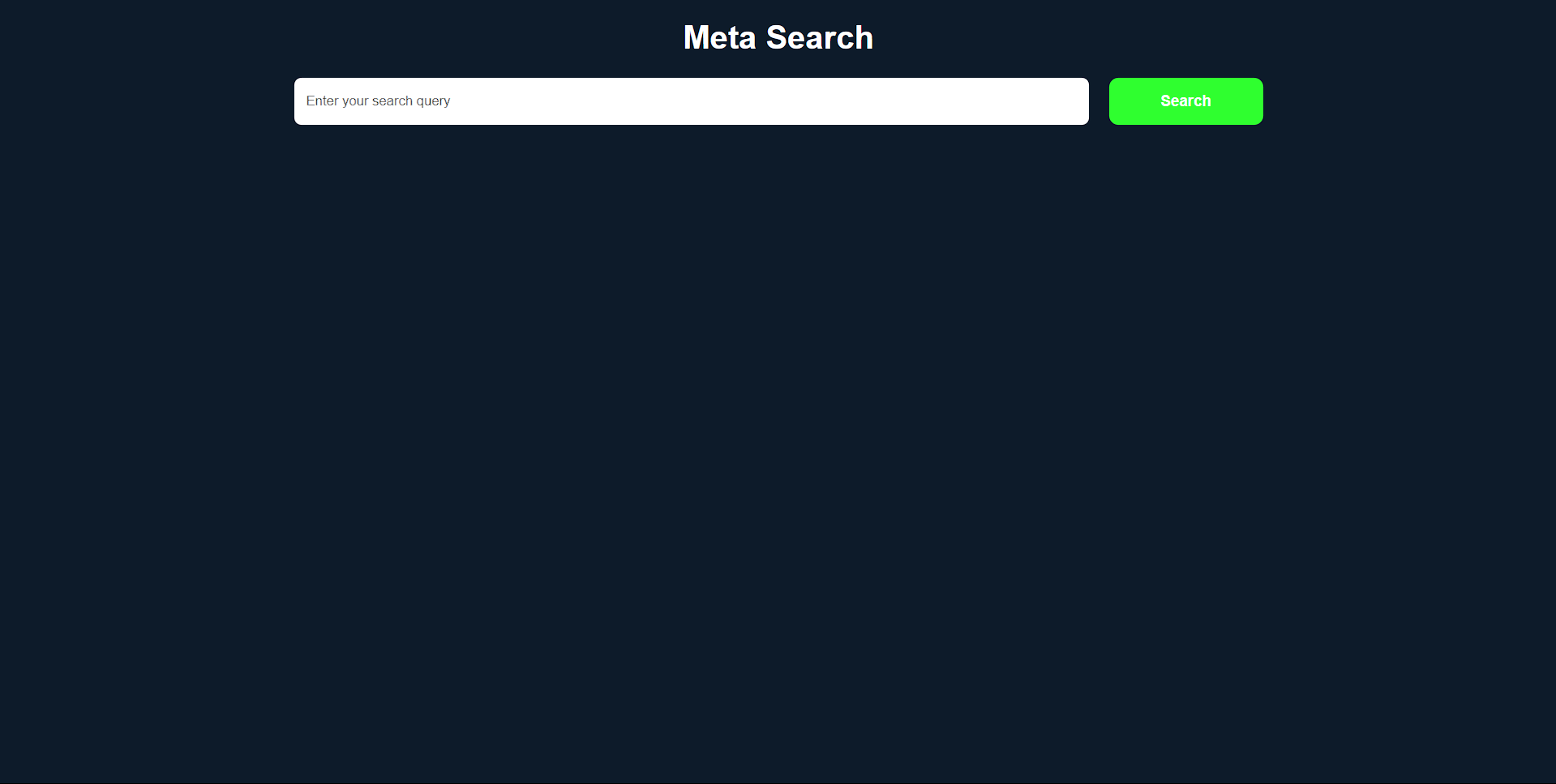
            results = await meta\_search(search\_query, num\_results=60)

    return render\_template("index.html", results=results)

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

    app.run(port=5001,debug=True)

**Screenshots :**



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.