TASK – Web Summarizer

Day – 4

📘 Objective:

The main objective of this project is to develop a web application that takes a website URL as input and provides the following:

1. ✅ A well-structured AI-generated summary using Google's Gemini API
2. ✅ A list of internal and external links from the webpage
3. ✅ A gallery of up to 30 images from the website
4. ✅ A clean, responsive frontend for ease of use
5. ✅ Optional typing effect for summary (like ChatGPT)

🧠 Learnings:

- How to use **Gemini API** for real-world NLP applications
- Parsing HTML with **BeautifulSoup**
- Creating a full-stack app using **Flask + JavaScript**
- Dynamic typing effects with **word-by-word rendering**
- Rendering Markdown securely in browser using markdown-it

Code:

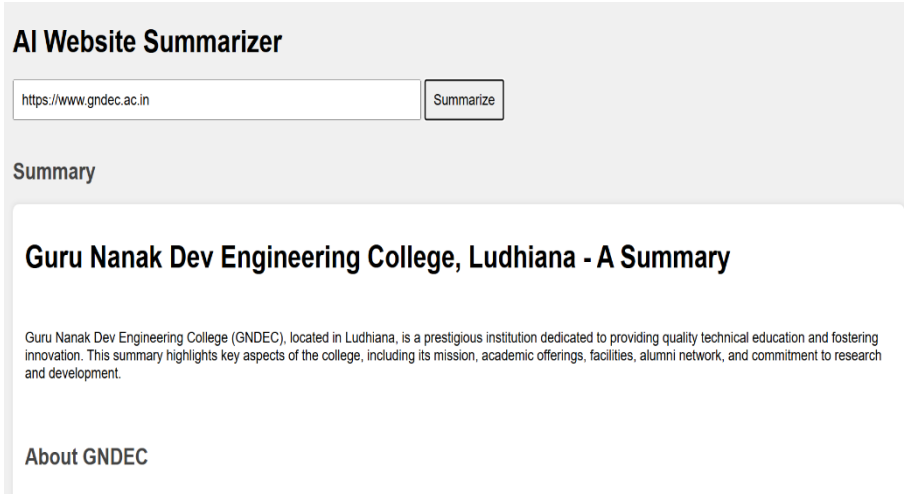
```
from flask import Flask, request, render_template, jsonify
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse, urljoin
import google.generativeai as genai
import markdown
# 🔑 Configure your Gemini API key
genai.configure(api_key="AIzaSy.....IQ0")
model = genai.GenerativeModel("gemini-2.0-flash")
```

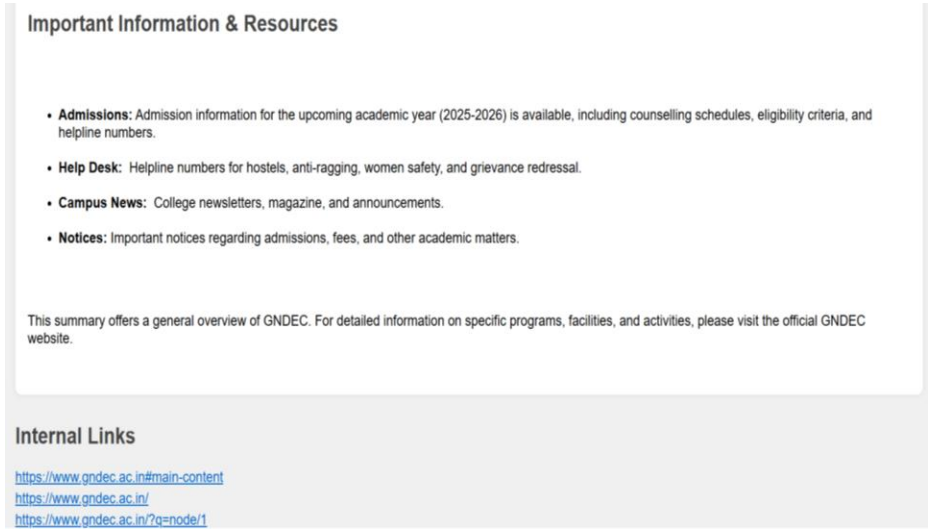
```
app = Flask(__name__)

def extract_clean_text(url):
    try:
        response = requests.get(url, timeout=30)
        soup = BeautifulSoup(response.text, "html.parser")
        for tag in soup(["script", "style", "img", "svg", "nav", "footer", "header", "aside", "noscript"]):
            tag.decompose()
        return soup.get_text(separator=" ", strip=True)[:10000]
    except Exception as e:
        return f"Error fetching content: {str(e)}"

def summarize_text(text):
    if not text.strip():
        return "❌ Sorry, I couldn't extract any readable content from the site. Please try another URL."
    try:
        prompt = f"Generate a beautiful summary of the given site in markdown with proper headings and structure.\n\n{text}"
        response = model.generate_content(prompt)
        return response.text.strip()
    except Exception as e:
        return f"Error summarizing: {str(e)}"

def extract_links(url):
    internal_links = set()
    external_links = set()
    try:
        response = requests.get(url, timeout=10)
        base_url = urlparse(url).netloc
        soup = BeautifulSoup(response.text, "html.parser")
```

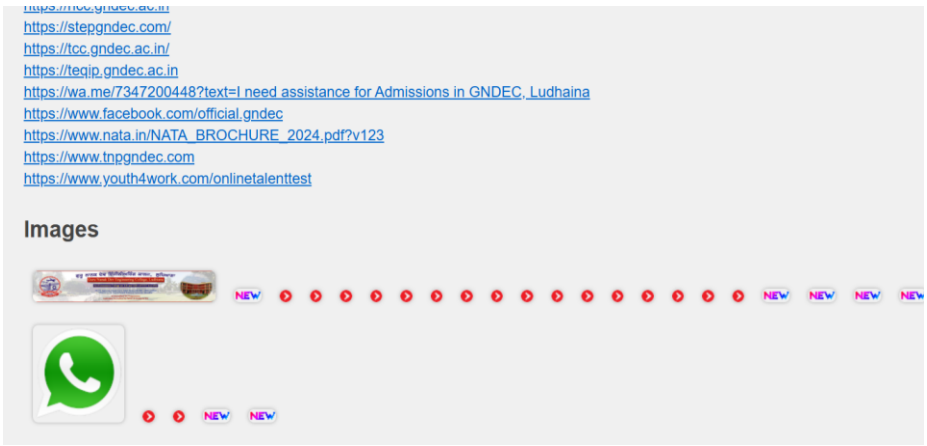
```
        for a_tag in soup.find_all("a", href=True):
            href = a_tag["href"]
            full_url = urljoin(url, href)
            parsed_url = urlparse(full_url)
            if parsed_url.netloc == "" or base_url in parsed_url.netloc:
                internal_links.add(full_url)
            else:
                external_links.add(full_url)
        return sorted(internal_links), sorted(external_links)
    except Exception as e:
        return [], []
def extract_images(url):
    image_links = []
    try:
        response = requests.get(url, timeout=10)
        soup = BeautifulSoup(response.text, "html.parser")
        for img_tag in soup.find_all("img"):
            src = (
                img_tag.get("src") or
                img_tag.get("data-src") or
                img_tag.get("data-original") or
                img_tag.get("data-lazy") or
                ""
            )
            if not src:
                continue
            full_url = urljoin(url, src)
            if full_url.lower().endswith(('.jpg', '.jpeg', '.png', '.gif', '.webp')):
                image_links.append(full_url)
```

```
        return image_links[:30]
    except Exception as e:
        return []
# ✅ This is the route that renders the main page
@app.route("/", methods=["GET"])
def index():
    return render_template("index.html")
# ✅ This route handles the AJAX request for streaming summary
@app.route("/generate_summary", methods=["POST"])
def generate_summary():
    url = request.json.get("url")
    text = extract_clean_text(url)
    markdown_summary = summarize_text(text)
    return jsonify({"summary": markdown_summary})
# ✅ This route gives internal/external links and images
@app.route("/get_details", methods=["POST"])
def get_details():
    url = request.json.get("url")
    internal_links, external_links = extract_links(url)
    image_links = extract_images(url)
    return jsonify({
        "internal_links": internal_links,
        "external_links": external_links,
        "images": image_links
    })
if __name__ == "__main__":
    app.run(debug=True)
```

Output:


AI Website Summarizer
https://www.gndec.ac.in
Summarize
Summary
Guru Nanak Dev Engineering College, Ludhiana - A Summary
Guru Nanak Dev Engineering College (GNDEC), located in Ludhiana, is a prestigious institution dedicated to providing quality technical education and fostering innovation. This summary highlights key aspects of the college, including its mission, academic offerings, facilities, alumni network, and commitment to research and development.
About GNDEC


Important Information & Resources
• Admissions: Admission information for the upcoming academic year (2025-2026) is available, including counselling schedules, eligibility criteria, and helpline numbers.
• Help Desk: Helpline numbers for hostels, anti-ragging, women safety, and grievance redressal.
• Campus News: College newsletters, magazine, and announcements.
• Notices: Important notices regarding admissions, fees, and other academic matters.
This summary offers a general overview of GNDEC. For detailed information on specific programs, facilities, and activities, please visit the official GNDEC website.
Internal Links
https://www.gndec.ac.in#main-content
https://www.gndec.ac.in/
https://www.gndec.ac.in/?q=node/1


https://stepgndec.com/
https://tcc.gndec.ac.in/
https://teqip.gndec.ac.in
https://wa.me/7347200448?text=I need assistance for Admissions in GNDEC, Ludhaina
https://www.facebook.com/official.gndec
https://www.nata.in/NATA_BROCHURE_2024.pdf?v123
https://www.tnpgndec.com
https://www.youth4work.com/onlinetalenttest
Images