DAY 4

Building Website Summarizer, Link Analyzer & Image Extractor Using Gemini AI

Today was one of the most hands-on and interesting days of my training. I got to build **three real-world tools** using Python and **Google's Gemini AI API** (Generative AI). These tools interact with live websites, fetch content, process it, and either summarize text, extract useful information, or even describe images. It really felt like combining AI with web scraping to make smart, automated utilities — and I'm excited to share everything I did today!

Website Summarizer using Gemini

The first tool I created was a **website summarizer**. The idea was simple: enter a URL, extract the main article or page content, and then generate a short, intelligent summary using the Gemini API.

How It Works:

- ❖ I started by writing a function to **fetch the website HTML** using requests and then **cleaned up the text** using BeautifulSoup. I removed unnecessary parts like <script>, <style>, headers, footers, navigation bars, etc.
- ❖ Once I got the clean and readable content, I passed it to **Gemini's model** (gemini-2.0-flash) with a prompt like:

“Please provide a concise summary of the following web article text. Focus on key points, main arguments, and conclusions.”

- The summary returned by Gemini was short and to the point — exactly 3 to 4 sentences, just as I instructed.
- I also added a function to **save the output to a file** (summary_output.txt) so that I can keep track of all summaries I generate.

Technologies Used:

- requests, BeautifulSoup – for scraping and cleaning the webpage
- google.generativeai – Gemini API for summarization
- dotenv – for safely managing API keys
- textwrap – for pretty printing text in the terminal and output files

Internal & External Link Analyzer + Multi-Link Summarizer

The second tool was an extension of the first — but more advanced. Instead of just summarizing one page, this tool analyzes **all the links** (internal and external) on a webpage and lets me **choose which links to summarize** one by one.

Key Features:

- The script starts by asking the user for a website URL.
- It extracts all <a> tags and separates them into **internal** (same domain) and **external** (different domain) links using Python’s urlparse and urljoin.

It then prints out all the links nicely, with numbering and labeling like:

1. [Internal] https://example.com/about

2. [External] https://wikipedia.org/some-topic

- ❖ After that, I could **type the number of the link** I wanted to summarize. The program fetches that page's text, passes it to Gemini for summarization, and appends the result to a file called summaries.txt.

What Made This Cool:

- ❖ I was able to summarize **multiple pages from one site**, just by selecting link numbers.
- ❖ This could actually be turned into an AI-powered browser assistant or research tool!

Website Image Extractor + Gemini Vision Describer

This one was my favorite. I created a tool that:

1. Extracts all images (<img>) from any website.
2. Download those images locally.
3. Uses **Gemini Vision model** to describe what each image shows (in natural language!).
4. Also detects the image format, size, and mode using the Pillow (PIL) library.

Step-by-Step:

- ❖ After entering a website URL, it scrapes all image sources from <img> tags.
- ❖ It constructs full URLs even for relative or protocol-relative paths (like /images/logo.jpg or //cdn.example.com/pic.png).

- Each image is downloaded and stored inside a folder (extracted_images).
- Then I use Gemini to **generate a caption or description** like:

  "A man holding a tablet while standing in front of a presentation screen."
- All image details, URLs, and descriptions are written into a report file (named like image_extract_20250714.txt), saved inside the same folder.

Real-Life Use:

- This could be super useful for **automated content auditing**, creating **accessibility descriptions** for images, or just analyzing visual content on the web using AI.

What I Learned Today

1. **How to interact with real-world websites using Python:**This included using headers to avoid being blocked, managing relative links, and cleaning raw HTML into readable text.
2. **How to use Google's Gemini API (Text & Vision models):**From summarizing text to understanding images, Gemini can do a lot with the right prompts.
3. **How to manage and structure a Python project** that combines scraping, AI, file handling, and user interaction.
4. **The power of combining scraping + AI**:Scraping gives you raw data. AI helps make sense of it — turning messy text and images into insights, summaries, and labels.

Tech Stack Recap

Tool/Library	Purpose
requests	Fetching HTML and images from websites

BeautifulSoup	Parsing and cleaning HTML
google.generativeai	Connecting to Gemini models (text + image)
dotenv	Storing API keys securely
PIL (Pillow)	Processing image files
textwrap,os	Utility handling (files, formatting)

Final Thoughts

Today's projects gave me a solid glimpse into what real-world AI-powered tools can look like. From summarizing articles to describing images and even analyzing links - all of it felt useful and practical.

More importantly, it wasn't just about using Gemini - it was about thinking how to **combine AI with existing Python tools** to solve actual problems. I'm now confident in building AI tools that work with real data from the web, and I'm excited to keep going!