NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(KARACHI CAMPUS)

FAST School of Computing

Spring 2025

Project Report

Web Crawler

Syeda Fakhira Saghir [22k-4413]

Aafreen [22k-4448]

Advisor: Sir Waseem rauf

TABLE OF CONTENT

1. Motivation

The motivation behind this project stems from the increasing need for automated data extraction and analysis from the vast amount of information available on the internet. With the exponential growth of web content, it becomes essential to develop intelligent tools that can crawl, collect, and analyze data efficiently. This project aims to simplify this process by creating a smart web crawler that is capable of retrieving and storing useful data in a structured database for future processing and visualization

2. Overview

2.1 Significance of the Project

This smart web crawler project is both practically and academically valuable. It provides an efficient solution to automate data extraction from websites, which is useful for research, marketing, SEO analysis, and more. The crawler not only fetches textual content and metadata but also stores images and hyperlinks, making it a versatile tool for web data mining. It highlights how a simple Python-based GUI can perform powerful data operations with SQLite, without requiring large-scale infrastructure.

2.2 Description of the Project

This project is a desktop-based graphical application developed using Python's Tkinter library. It allows the user to input a URL, crawl that site and its internal links, extract titles, hyperlinks, and images, and store them in an SQLite3 database. The application shows real-time crawling progress, supports viewing of the visited URLs, and provides database statistics (total pages, links, and images collected). The goal is to offer an easy-to-use tool for structured web crawling and data collection.

2.3 Background of the Project

The project is based on the foundational concepts of web crawling and scraping. Libraries like requests, Beautifulsoup, and sqite3are used extensively. The GUI is developed using Python's tkinter. Previous works in this domain include web scraping tools like Scrapy, but our project focuses on a lightweight, GUI-based crawler ideal for small-scale applications.

References:

- BeautifulSoup Documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Python requests library: https://docs.python-requests.org/
- SQLite documentation: https://sqlite.org/docs.html

2.4 Project Category:

This is a Product-Based project.

3. Features / Scope / Modules

1. **Graphical User Interface**
   - A clean and user-friendly interface built with Tkinter.
   - Allows users to input the starting URL.

2. **Smart Web Crawling**
   - Automatically crawls internal links within the specified domain.
   - Skips duplicate and invalid links.

3. **Data Extraction and Storage**
   - Extracts page titles, hyperlinks, and image URLs.
   - Stores data in a structured format using SQLite3.

4. **Database Status Viewer**
   - View number of pages, links, and images crawled.

5. **Visited URLs Viewer**
   - Displays all the URLs that have been visited during crawling.

Libraries Used and Their Purposes

1. **requests** - For making HTTP requests to fetch web pages
2. **BeautifulSoup (bs4)** - For parsing HTML content and extracting data
3. **urllib.parse** - For URL manipulation and joining relative URLs
4. **sqlite3** - For storing crawled data in a local database
5. **threading** - For concurrent crawling operations
6. **tkinter** - For the graphical user interface
7. **selenium** - For rendering JavaScript-heavy pages
8. **collections.Counter** - For counting keyword frequencies
9. **re** - For regular expression pattern matching
10. **textblob** - For sentiment analysis of page content
11. **concurrent.futures** - For thread pool management
12. **requests.adapters/Retry** - For request retry logic
13. **json** - For serialization data for storage
14. **os** - For potential file system operations
15. **PIL (Image, ImageTk)** - For image processing in the GUI
16. **io.BytesIO** - For handling binary image data

Functions

1. setup_theme - Defines color scheme for the GUI
2. setup_requests_session - Configures the HTTP session with retry logic
3. setup_driver_options - Sets options for Selenium WebDriver
4. setup_gui - Creates and arranges all GUI components
5. get_dynamic_html (static) - Fetches JavaScript-rendered pages using Selenium
6. analyze_text_content (static) - Analyzes text content for keywords and sentiment
7. is_valid_url - Validates URL format and scheme
8. create_tables - Initializes the database schema
9. crawl_page - Main crawling loop that manages the crawling process
10. process_page - Processes an individual page (fetching, parsing, storing)
11. check_robots_txt - Checks robots.txt for crawling permissions
12. store_page_data - Stores page metadata in the database
13. store_links - Stores discovered links in the database
14. store_images - Stores image information in the database
15. start_crawling_thread - Starts the crawling process in a separate thread
16. stop_crawling - Stops an active crawling process
17. toggle_buttons - Enables/disables GUI buttons based on crawler state
18. update_current_url - Updates the GUI with the currently processed URL
19. update_status - Updates the status bar in the GUI
20. store_data - Legacy method for storing data (partially redundant)
21. log_message - Logs messages to the output box
22. log_error - Logs error messages to the output box
23. view_db_status - Displays database statistics in the GUI
24. view_visited_urls - Displays all visited URLs in the GUI

4. Project Planning

Research & Setup [Week 1]

- Studied web crawling uses, techniques and tools.
- Set up the development environment for Python.
- Planned project structure and technologies.
- Distributed work.

Core Crawling Development [Week 2]

- Developed basic web crawlers using Scrapy.
- Implemented page fetching and parsing of url.
- Set up request handling with Requests and Selenium.

Data Extraction & Storage [Week 3]

- Ensured extraction of useful content using BeautifulSoup and Scrapy.
- Design and set up a database in sqlite for storing crawled data.
- We Implemented indexing for efficient search and retrieval.

Analysis & Optimization [Week 4]

- Added keyword and data analysis.
- Handled duplicate urls.
- Handled errors in url.
- Worked on improving gui.

Testing & Finalization [Week 5]

- Performed testing to ensure functionality.
- Document project setup and usage.
- Prepared a project report and recorded a demo.

5. Project Feasibility

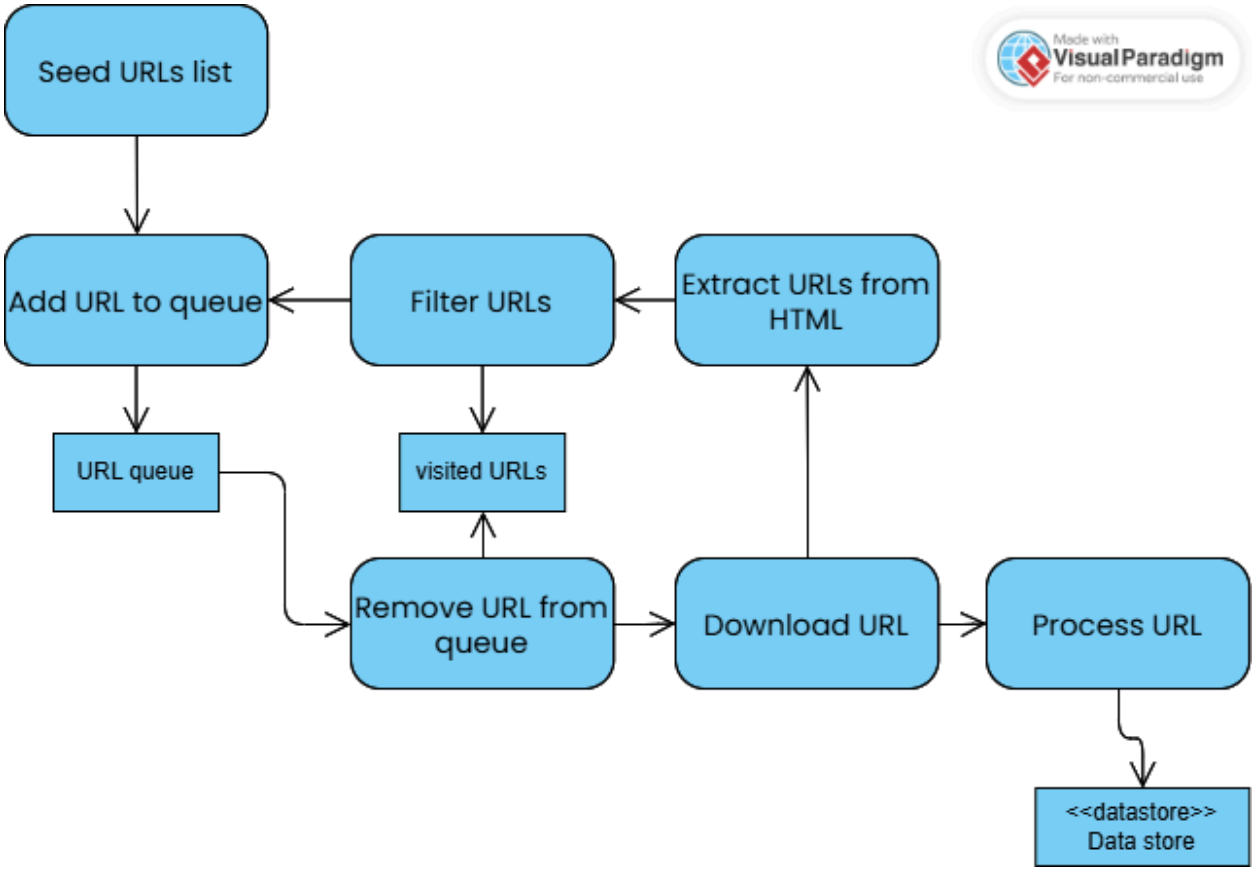
- **Technical Feasibility:** The project uses technologies that are widely available and well-supported in Python. No external APIs or third-party servers are required.
- **Economic Feasibility:** The project is free to develop using open-source tools. It requires no external investment and can run on any standard PC.
- **Schedule Feasibility:** The project is small and manageable, and the proposed schedule ensures completion within 8 weeks.

6. Hardware and Software Requirements

- **Hardware:**
  - Minimum 4GB RAM, 500MB Disk Space, x86 or x64 processor.

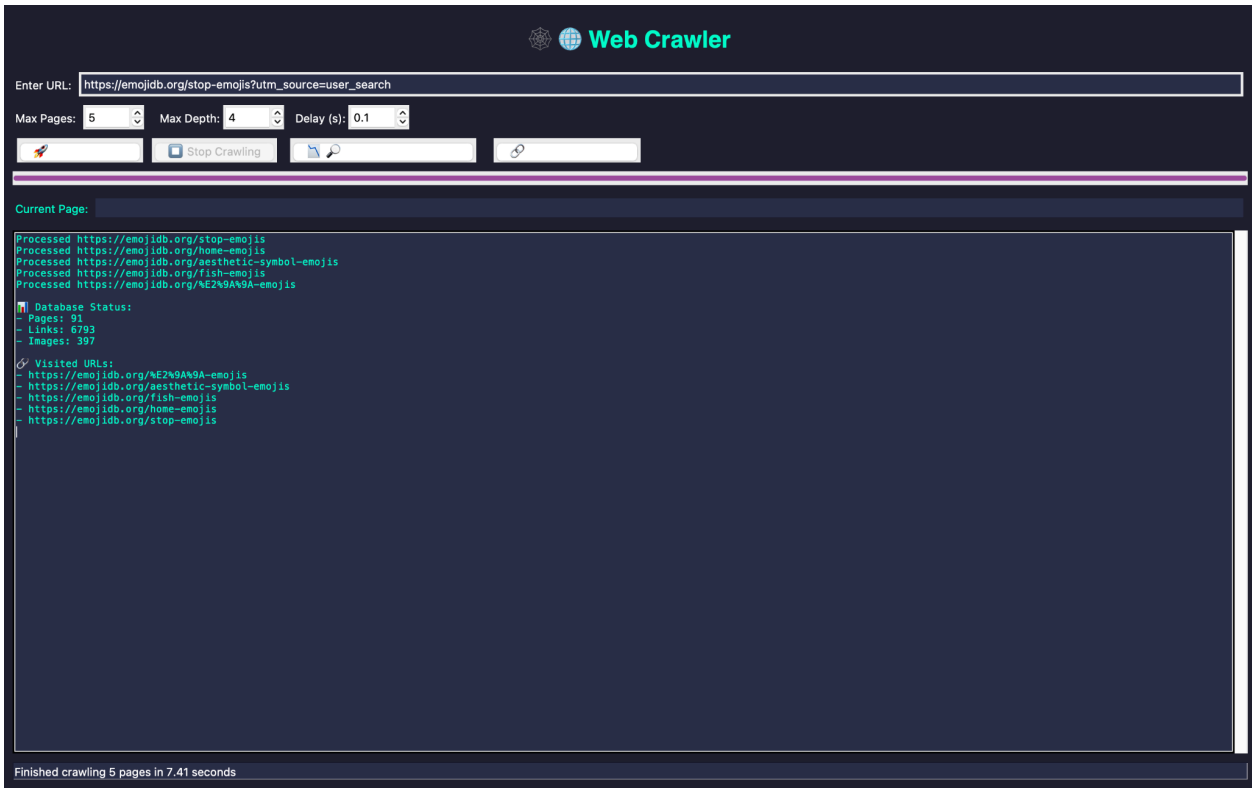
- **Software:**
  - OS: Windows/Linux/Mac
  - Python 3.8 or above
  - Required Libraries: request, tkinter, bs4, sqlite3, urllib.parse etc.

7. Diagrammatic Representation of the Overall System

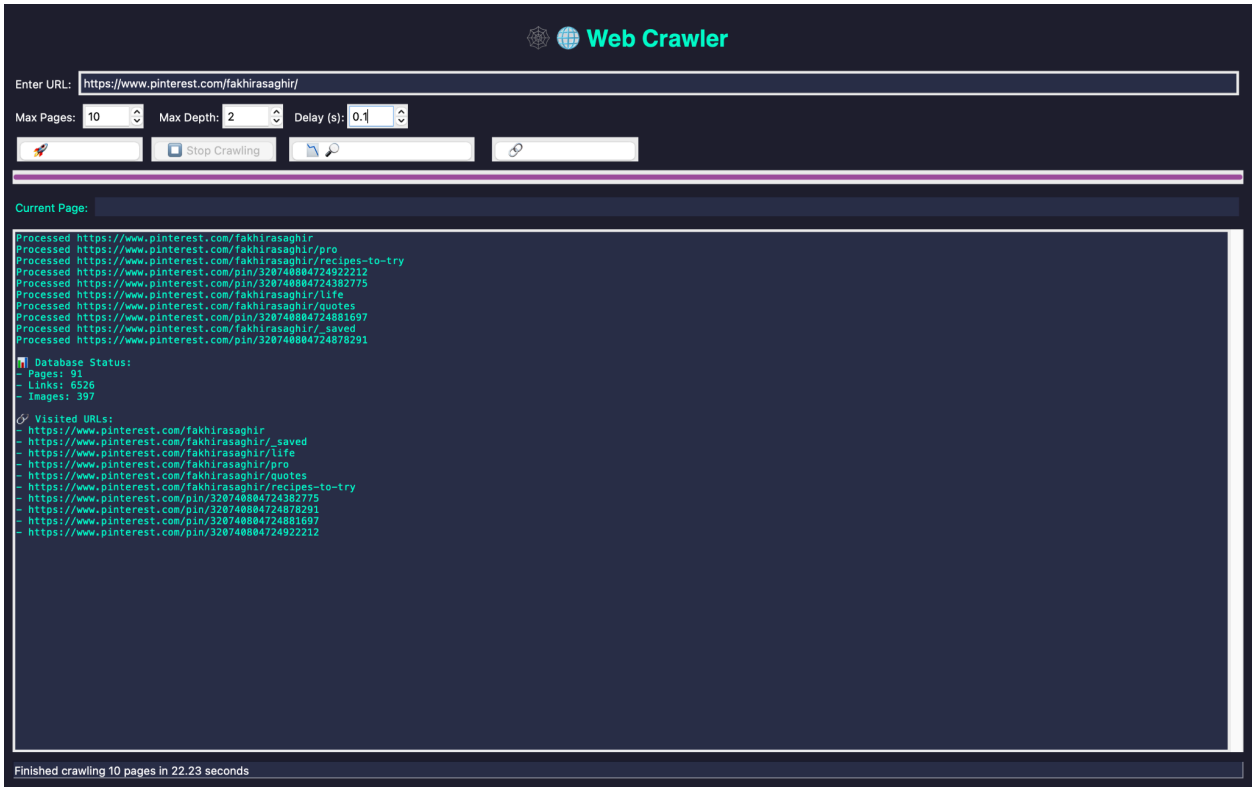


OUTPUT:

Crawl:
https://emojidb.org/stop-emojis?utm_source=user_search



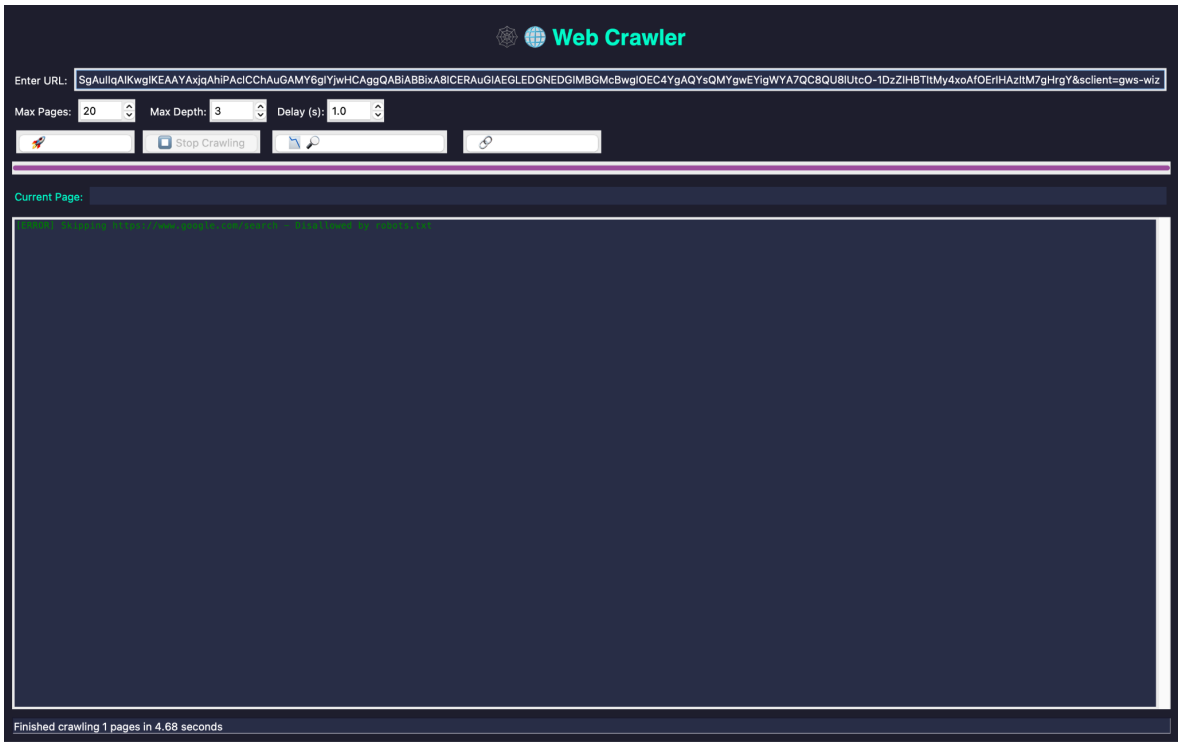
Crawl:
https://www.pinterest.com/fakhirasaghir/



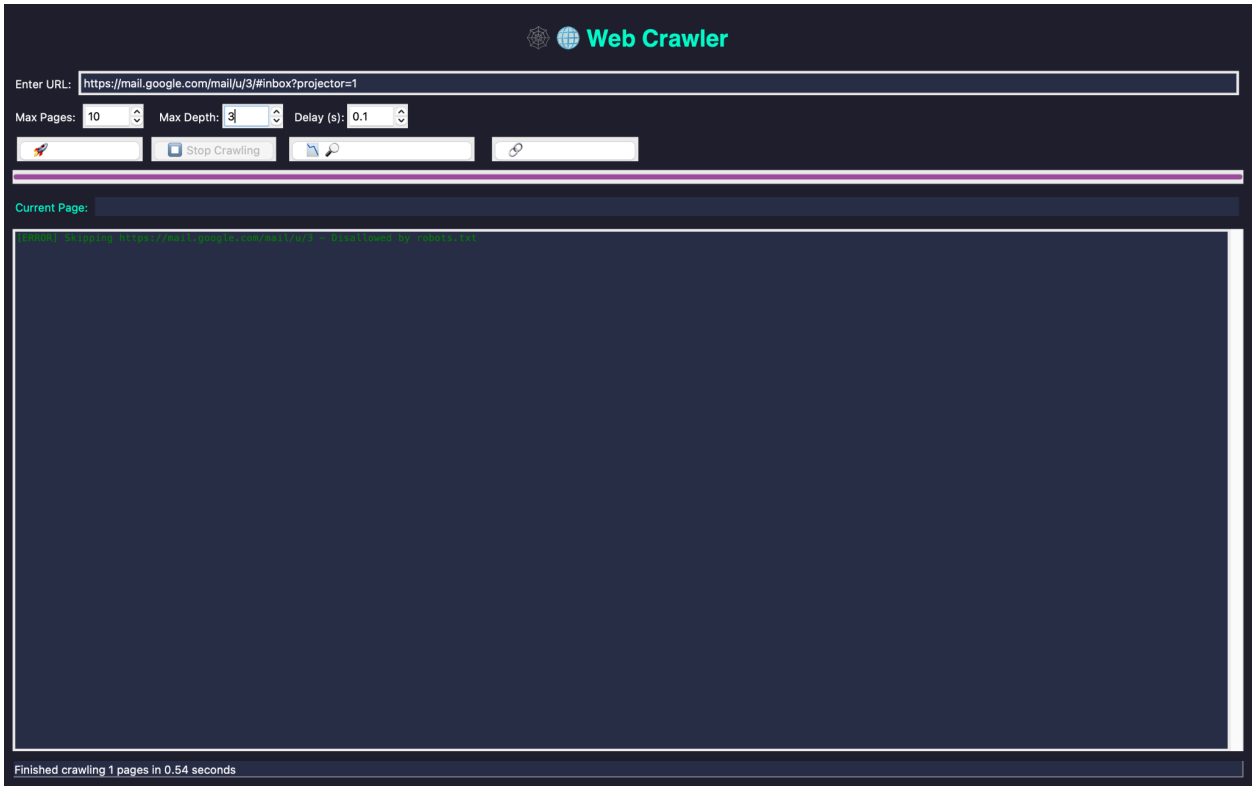
Crawl:

https://www.google.com/search?q=web+crawling&sca_esv=b5608bee10f15177&authuser=2&source=hp&ei=3s0MaN-4L7eVxc8Px_jl-As&iflsig=ACkRmUkAAAAAaAzb7vIQwVfz26R4BzVyBovDpBHUkqZM&oq=web&gs_lp=Egdnd3Mtd2l6IgN3ZWIqAggBMgsQABiABBiRAhiKBTILEAAYgAQYkQIYigUyChAAGIAEGEMYigUyChAAGIAEGEMYigUyDRAAGIAEGLEDGEMYigUyChAAGIAEGEMYigUyEBAAGIAEGLEDGEMYgwEYigUyChAAGIAEGEMYigUyEBAAGIAEGLEDGEMYgwEYigUyChAAGIAEGEMYigVInihQjhNYzxZwAXgAkAEAmAGKAqAB8gWqAQMyLTO4AQPIAQD4AQGYAgSgAuIIqAIKwgIKEAAYAxjqAhiPAcICChAuGAMY6gIYjwHCAggQABiABBixA8ICERAuGIAEGLEDGNEDGIMBGMcBwgIOEC4YgAQYsQMYgwEYigWYA7QC8QU8lUtcO-1DzZIHBTItMy4xoAfOErIHAzItM7gHrgY&sclient=gws-wiz

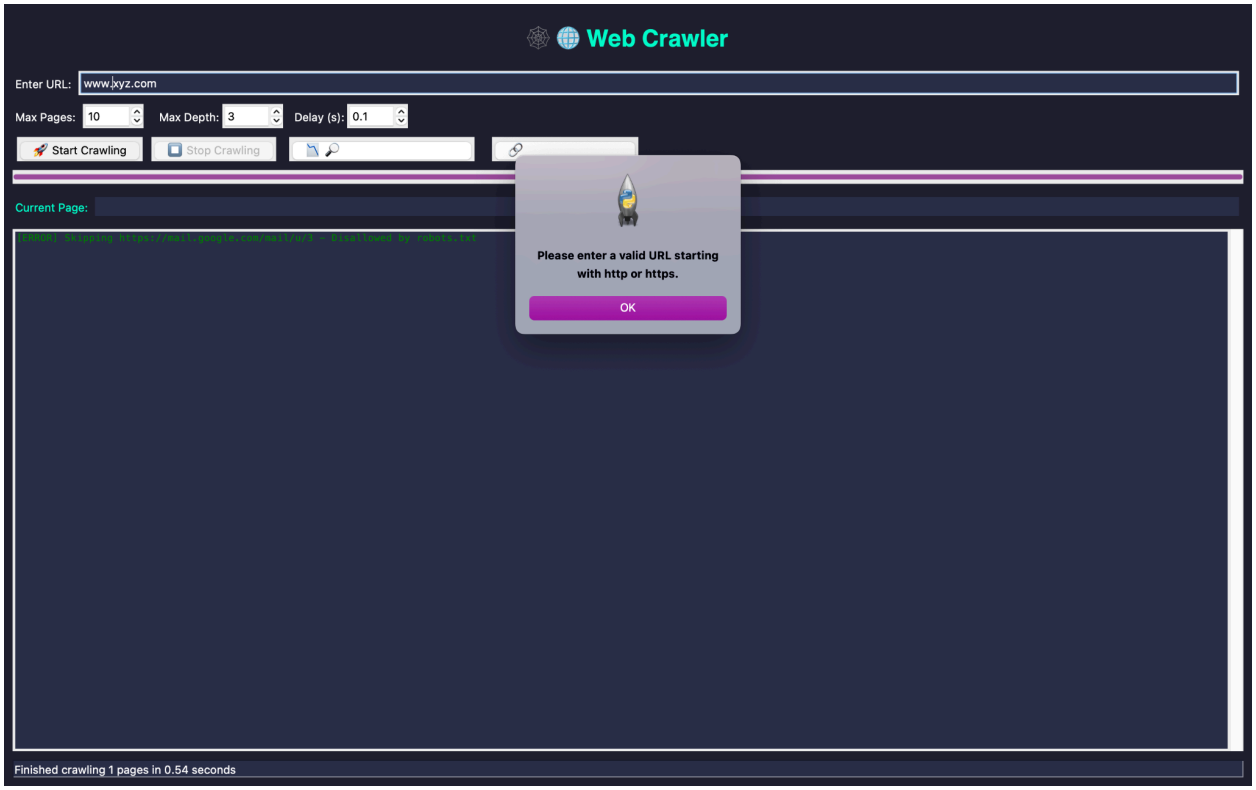
Already visited so crawler skipped this website



https://mail.google.com/mail/u/3/#inbox?projector=1



Crawl:
www.xyz.com (wrong url)


Web Crawler
Enter URL:
www.xyz.com
Max Pages:
10
Max Depth:
3
Delay (s):
0.1
Start Crawling
Stop Crawling
Current Page:
Please enter a valid URL starting with http or https.
OK
Finished crawling 1 pages in 0.54 seconds