**Web Crawler for Recipe Extraction from AllRecipes**

**1. Introduction**

This project implements a Python-based web crawler to extract cooking recipes from the [AllRecipes](https://www.allrecipes.com/) website. The script fetches recipe titles, content (like instructions and ingredients), and associated images, and stores the results in a structured text file. It also visualizes the most frequently occurring recipe titles and displays the corresponding images.

**2. Objectives**

* Automatically crawl and extract cooking recipes from a designated domain.
* Retrieve and organize recipe information: title, instructions, and image.
* Save the data in a readable text file.
* Provide data visualization for popular recipes.
* Display downloaded recipe images.

**3. Technologies Used**

* **Python**: Main programming language.
* **Libraries**:
  + requests: For sending HTTP requests.
  + BeautifulSoup (from bs4): For HTML parsing.
  + collections: For managing visited URLs and counting titles.
  + re, time, deque: Standard libraries for regex, delays, and queue management.
  + matplotlib.pyplot: For generating bar charts.
  + PIL (Python Imaging Library): For image processing and display.

**4. Core Features**

**4.1 Crawler Configuration**

* START\_URLS: List of initial seed URLs.
* ALLOWED\_DOMAIN: Restricts crawling to AllRecipes only.
* OUTPUT\_FILE: File path to save the extracted recipes.

**4.2 Valid URL Check**

def is\_valid\_recipe\_url(url):

**4.3 Link Fetching**

def fetch\_links(url):

**4.4 Recipe Extraction**

def extract\_recipe\_title\_and\_content(url):

**4.5 Crawling Logic**

def crawl\_recipes(start\_urls, max\_recipes):

* Breadth-First Search approach using deque.
* Recursively crawls valid recipe URLs until the desired number of recipes is collected.
* Stores each recipe in a dictionary: {url: (title, content, image\_url)}.

**4.6 Data Output**

def save\_to\_txt(data, filename):

Writes the structured content into a text file.

**5. Visualization**

**5.1 Title Frequency Chart**

Counter(title for title, \_, \_ in results.values())

* Counts frequency of recipe titles.
* Displays the top 15 using a horizontal bar chart (matplotlib).

**5.2 Image Display**

Image.open(BytesIO(response.content))

* Downloads and displays each recipe’s main image (if available).
* Uses PIL to show the image with matplotlib.

**6. User Interaction**

* Prompts the user for the number of recipes to crawl.
* Handles invalid inputs gracefully using exception handling (try-except).

**7. Error Handling**

* Catches exceptions during HTTP requests, HTML parsing, and image downloads.
* Ensures the crawler continues even if individual pages fail.

**8. Sample Output Format (Text File)**

Spaghetti Carbonara

https://www.allrecipes.com/recipe/12345/spaghetti-carbonara/

=== Ingredients ===

- 2 eggs.

- 1/2 cup grated cheese.

...

=== Instructions ===

1. Cook pasta.

2. Whisk eggs and cheese.

...

==================================================

**9. Potential Improvements**

* Limit depth or page count to avoid crawling loops.
* Add logging instead of print() for production use.
* Store data in JSON or a database for easier reuse.
* Improve image selection with more precise heuristics.
* Handle JavaScript-loaded content (e.g., using Selenium).

**10. Conclusion**

This project effectively demonstrates web scraping, content parsing, and data visualization for recipe collection. It is a robust and extensible base for building food-related applications, recipe recommendation engines, or culinary datasets.