Documentation for Ngmc Search Engine Web Application

Overview

This documentation provides a comprehensive guide to the Bing Search Engine web application, which is built using Flask and integrates with Bing's search functionality. The application allows users to search the web using Bing and displays the results in a dark-themed user interface resembling GitHub's dark mode.

Key Components

1. **HTML Template** (`search.html`)
2. **CSS Styles** (embedded in `<style>` tag)
3. **Flask Application** (`app.py`)
4. **Bing Search Integration**

1. HTML Template (`search.html`)

The `search.html` file defines the structure and layout of the web application's front end. It includes:

Head Section:
  Meta Tags: Define character encoding and viewport settings for responsive design.
  Google Fonts: Import fonts to be used in the application.
  Title: Sets the page title.
  Stylesheet Link: Links to an external CSS file (if present).

Body Section:
  Container:The main wrapper for content with a dark background.
  Search Form: Allows users to input search queries and submit them.
  Search Results: Displays search results if a query is provided.

Embedded CSS:
Body Styles: Sets a dark background and light text color to mimic GitHub's dark mode.
Container Styles: Provides styling for the main content area.
Form Styles: Styles the search input and button with dark mode colors.
Result Styles: Formats the search results with dark backgrounds and light text.
Responsive Styles: Ensures the layout adapts to different screen sizes.

2. CSS Styles

The CSS embedded within the `<style>` tag provides a GitHub dark mode-inspired appearance:

Body and Container:** Uses dark colors for background and text to align with GitHub's dark mode.
Text and Input Fields:** Light text colors on dark backgrounds for readability.
Form Buttons:** Styled to stand out with a GitHub-like green color.

Results List: Dark background with light text, with hover effects to enhance interactivity.
Responsive Design: Adjusts layout and font sizes for various screen sizes to ensure usability on mobile devices.

3. Flask Application (`app.py`)

The Flask application provides the backend functionality of the web app:

Imports:
  Flask: Web framework for handling requests and rendering templates.
  Requests: For making HTTP requests to Bing.
  BeautifulSoup: For parsing HTML responses from Bing.

`bing_search(query, num_pages=20)` Function:
  Purpose: Retrieves search results from Bing.
  Parameters:
  query`: Search query string.
  um_pages`: Number of pages to retrieve (default: 20).
  Returns: List of search result dictionaries containing `title`, `link`, and `snippet`.

`home()` Route:
  Method: Handles both GET and POST requests.
  Functionality:
   Displays the search form on GET requests.
   Processes the search query on POST requests, retrieves results, and renders them using the `search.html` template.

Running the Application:
 The application is set to run in debug mode for development purposes.

4. Bing Search Integration

Search URL Construction: Uses Bing's search URL format to query results.
Result Parsing: Extracts titles, links, and snippets from search results.
Error Handling: Logs errors if Bing's response is unsuccessful.
Pagination: Retrieves multiple pages of results as specified.

Example Usage

1. Access the Application:
    Navigate to `http://localhost:5000` in a web browser.

2. Perform a Search:
   Enter a search query into the input field and click "Search."
   View the results displayed below the search form.

3. Responsive Design:
   Test on various devices to ensure the layout adjusts appropriately.

Conclusion

This documentation covers the essential aspects of the Bing Search Engine web application, including its structure, styles, backend functionality, and integration with Bing's search API. For further customization or additional features, you may modify the HTML, CSS, and Flask code as needed.