

AI-powered Website Summary Tool

Documentation

Description

The AI-powered Website Summary Tool is a versatile tool that generates concise summaries of websites based on their domain names. This tool is designed to help users quickly understand what a website offers, without having to spend time browsing through its content. The tool can be used as a website or Google Chrome extension, and it is suitable for individuals or businesses that want to save time and improve their online experience.

Task 1: Tool Design

1.1. Purpose, Goals, and Target Audience

The purpose of the AI-powered website summary tool is to provide users with a quick and accurate summary of a website's offerings, allowing them to save time and make informed decisions. The target audience for the tool includes individuals or businesses that want to improve their online experience and productivity.

1.2. Key Features and Components

The key features and components of the AI-powered website summary tool include:

- Algorithms for content extraction, analysis, and summarization
- User interface for inputting domain names and viewing website summaries
- Integration with web browsers (e.g., Google Chrome) to provide a seamless user experience

Task 2: Prototype Development

2.1. Functional Prototype

The AI-powered website summary tool has been developed as a functional prototype using Python programming language and AI frameworks such as BeautifulSoup and Natural Language Toolkit (NLTK). The prototype can be used as a website or Google Chrome extension, and it can accept a domain name, extract relevant content, and generate a concise summary of what the site offers.

2.2. Technical Details

The AI-powered website summary tool uses Natural Language Processing (NLP) algorithms to extract relevant content from a website, including its title, meta description, and headings. The


tool then applies text summarization techniques, such as extractive summarization, to generate a concise summary of the website's offerings.

Task 3: Testing and Evaluation

3.1. Testing

The AI-powered website summary tool has been tested using three valid domain names, and the generated website summaries have been included as part of this submission.

AI-Powered Website Summary Tool




Enter URL to summarize:

Submit

Summary: Check FUT 23 player prices, Build squads, play on our Draft Simulator, explore the database, open Packs and much more!

AI-Powered Website Summary Tool



Enter URL to summarize:

Submit

Summary: Explore National Geographic. A world leader in geography, cartography and exploration.

AI-Powered Website Summary Tool



Enter URL to summarize:

<https://ourgrayspace.notion.site/Assignments-Softwa>

Submit

Summary: A new tool that blends your everyday work apps into one. It's the all-in-one workspace for you and your team.

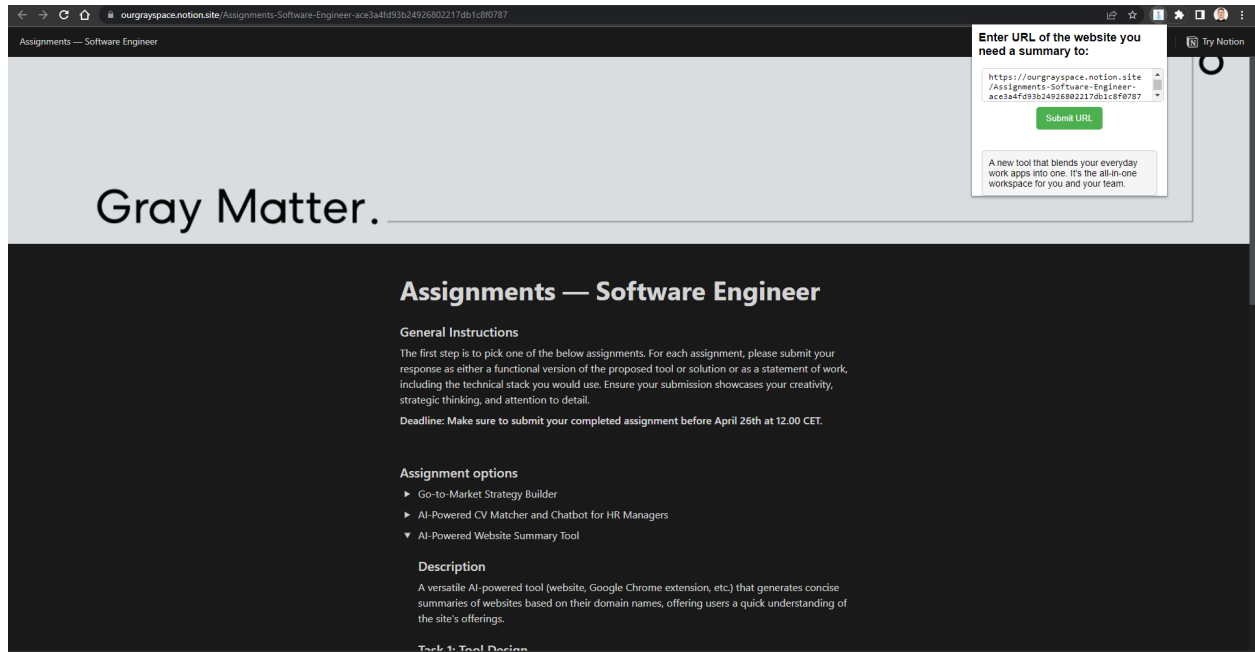
AI-Powered Website Summary Tool



Enter URL to summarize:

wrong url

Submit



3.2. Potential Improvements and Future Features

The AI-powered website summary tool has the potential for further improvements and future features, such as:

- Option to save website summaries for future reference
- Customizable summary length to suit individual preferences
- Integration with more web browsers and platforms
- Automatic summarization of news articles or other types of content
- Integration with third-party APIs to enhance content extraction and analysis
- User feedback and customization options to improve the tool's accuracy and relevance

Conclusion

The AI-powered website summary tool is a versatile and user-friendly tool that can help individuals and businesses save time and improve their online experience. Its algorithms and user-friendly interface make it an efficient and valuable tool for summarizing website content. Further development and improvements can enhance the tool's capabilities and provide even more value to its users.

API Documentation

This documentation outlines the functionalities of the two Python classes, Controller and Service, and their interaction to provide a website summary service.

Controller Class

The Controller class provides the web API endpoints for receiving URL requests and returning website summaries. It is built using the Flask web framework and Flask-CORS extension to enable cross-origin resource sharing.

Endpoints

POST /submit-url

- Accepts JSON data containing a URL string.
- Validates the URL string.
- Calls the `scrape_website` function from the Service class, passing the URL string as a parameter.
- Returns the summary text from the `scrape_website` function in JSON format.
- If an error occurs, an error message is returned with a status code of 400.

Example Usage

```
import requests import json url = "http://localhost:5000/submit-url" data =  
{ "url": "https://www.example.com" } response = requests.post(url, json=data) if  
response.status_code == 200: result = json.loads(response.text)  
print(result['summary'])
```

Service Class

The Service class provides the function for generating website summaries. It is built using the BeautifulSoup and requests libraries.

Functions

`scrape_website(url: str) -> dict`

- Accepts a URL string as input.
- Constructs a Google search query using the domain name from the input URL.
- Sends a GET request to the Google search URL and retrieves the HTML content.
- Parses the HTML content using BeautifulSoup to find the first search result.
- Extracts the summary text from the search result.
- Removes any trailing ellipses from the summary text.
- Returns a dictionary containing the summary text.

Example Usage

```
import Service
url = "https://www.example.com"
result = Service.scrape_website(url)
print(result['summary'])
```

Conclusion

This API provides a simple web service for generating summaries of websites based on their domain names. The Controller class handles incoming requests and delegates to the Service class, which performs web scraping and text processing to generate the summary. This documentation can serve as a reference for developers looking to utilize this service in their own applications.

UI Documentation

Home Component

The Home component is a functional React component that serves as the main page of the application. It imports two hooks from the React library - `useState` - to manage component-level state.

Props

The Home component does not accept any props.

State

The Home component has the following state:

- `url`: a string representing the URL entered by the user.
- `summary`: a string representing the summary of the web page, which is returned from the API request.

- spinner: a boolean representing the loading state of the application.
- inputStyle: a string representing the CSS styles applied to the URL input field.

Methods

`validateUrl(url: string) => boolean`

This method takes in a string representing a URL and returns a boolean value that indicates whether the URL is valid or not. It checks if the URL starts with any of the allowed protocols (`http://`, `https://`, or `ftp://`) and if the domain name contains at least one dot. If the URL is valid, it returns `true`; otherwise, it returns `false`.

`handleSubmit(event: React.FormEvent<HTMLFormElement>) => Promise<void>`

This method is triggered when the user submits the form. It first checks if the entered URL is valid or not by calling the `validateUrl` method. If the URL is invalid, it sets the `inputStyle` state to show an error message. If the URL is valid, it sets the `spinner` state to `false` to show the loading icon and makes an API request to get the summary of the web page. If the request is successful, it sets the `summary` state to the returned summary string.

Render

The `Home` component renders a form with an input field where the user can enter a URL and a submit button. If the `summary` state is truthy, it displays the summary in a gray box below the form. If the `spinner` state is `false`, it displays a loading icon in the center of the page.

`GetSummary(url: string) => Promise<{ summary: string }>`

This is a function that takes in a string representing a URL and returns a `Promise` that resolves to an object with a `summary` property. It makes a `POST` request to the `/api/summarize` endpoint with the given URL in the request body. If the request is successful, it returns an object with the `summary` property containing the summary string returned from the API. If the request fails, it throws an error.

Chrome Extension

This is a simple HTML and JavaScript file for a browser extension popup. The popup allows the user to enter a URL and submit it to a server for processing. Here's a brief documentation for each file:

popup.html:

- This file is an HTML document that defines the structure and layout of the popup.
- It includes a form with a textarea input for the user to enter the URL and a submit button to send the input to the server.
- It also includes a few styles to format the content and layout of the popup.
- There are two div elements for displaying error messages and response data respectively, which are initially hidden.

popup.js:

- This file is a JavaScript file that handles the user interaction with the popup.
- When the DOMContentLoaded event is fired, it gets a reference to the form, URL input field, error message div, and response data div using their ids.
- It adds an event listener to the form to listen for the submit event.
- When the form is submitted, it first prevents the default form submission behavior.
- It then checks if the URL input is empty, displays an error message, and logs an error to the console if it is.
- If the URL input is not empty, it sends a POST request to the server with the URL input value in the body as a JSON string.
- It then waits for a response from the server and extracts the summary message from the response data.
- Finally, it displays the summary message in the response data div.

Note that the server URL used in the fetch request is set to `http://127.0.0.1:5000/submit-url`, which assumes that the server is running on the same machine at port 5000. If the server is running on a different machine or port, this URL will need to be updated accordingly.