youtube\_transcript\_api(library)

The youtube\_transcript\_api is a Python library that allows you to retrieve video transcripts (subtitles) from YouTube. It can be used to fetch the captions of YouTube videos, whether they are automatically generated or manually uploaded by the video creator.

This library makes it easy to access the transcript of any publicly available YouTube video, provided that the video has captions enabled. It works by interacting with YouTube's internal APIs and extracting the available transcript data.

1. **Retrieve Transcripts:** You can retrieve the transcript for a YouTube video by passing the video's URL or video ID.
2. **Support for Multiple Languages:** It supports retrieving captions in different languages, if available.
3. **Automatic and Manual Captions:** It works with both automatic captions generated by YouTube's speech recognition system and captions uploaded by the video creator.
4. Easy to use:The api is relatively simple to use ,making it is a great tool for developers who need to analyze or use the text content of videos

You can install it using pip:

pip install youtube-transcript-api

Here's an example of how you can use the library to fetch the transcript of a video:

from youtube\_transcript\_api import YouTubeTranscriptApi

# Replace with the YouTube video ID

video\_id = 'dQw4w9WgXcQ'

# Fetch the transcript

transcript = YouTubeTranscriptApi.get\_transcript(video\_id)

# Print the transcriptfor entry in transcript:

print(f"Time: {entry['start']} - Text: {entry['text']}")

Common Methods:-

* get\_transcript(video\_id): Retrieves the transcript of the video.
* get\_transcripts([video\_ids]): Retrieves transcripts for multiple videos.
* get\_available\_languages(video\_id): Gets the available languages for a video's transcript.
* download\_transcript(video\_id, lang='en'): Downloads the transcript in a specified language (default is English).

This tool is useful for developers working on projects involving video content analysis, such as creating search engines for YouTube, processing video data for sentiment analysis, or automatically generating summaries.

Streamlit Concepts that are used in this project:

1.st.title():

* **Description:** This sets the title of your Streamlit web page.
* **In this code:** It sets the title of the app as "YouTube Transcript Summarizer".

st.title('YouTube Transcript Summarizer')

2.st.text\_input():

* **Description:** This creates a text input box in the Streamlit UI where users can type in data. In this case, it's used to input the YouTube video URL.
* **In this code:** It asks the user to enter a YouTube video link.

youtube\_link = st.text\_input('Enter YouTube video link:')

3.st.image():

* **Description:** This displays an image on the Streamlit app. Here, it displays the thumbnail of the YouTube video, which is fetched dynamically based on the video ID.
* **In this code:** It shows the thumbnail of the YouTube video if the URL is entered by the user.

if youtube\_link:

video\_id = youtube\_link.split("=")[-1]

st.image(f"http://img.youtube.com/vi/{video\_id}/0.jpg", use\_container\_width=True)

4.st.button():

* **Description:** This creates a button that the user can click to trigger an action. In this case, it is used to trigger the process of fetching the transcript and generating the summary.
* **In this code:** It shows a button labeled "Get Detailed Notes". When clicked, it fetches the transcript, generates the summary, and displays the results.

if st.button('Get Detailed Notes'):

transcripted\_text = extracted\_transcript\_detail(youtube\_link)

5.st.markdown():

* **Description:** This displays markdown content in the Streamlit app. Markdown is a lightweight markup language with plain-text formatting syntax.
* **In this code:** It uses st.markdown("### Detailed Notes:") to create a heading that labels the area where the summary will be shown.

st.markdown("### Detailed Notes:")

6.st.write():

* **Description:** This function is used to display output, such as text, data frames, or other objects. It auto-detects the type of the input (text, dataframe, plot, etc.) and displays it accordingly.
* **In this code:** It displays the AI-generated summary under the section labeled "Detailed Notes."

st.write(summary.content)