**PROJECT REPORT**

ON

**YouTube Video Summarizer**

**(CSE VI Semester Mini Project 2024-25)**



Submitted To: Submitted By:

Ms. Himadri Vaidya Brahm Sagar Gaur

(CC-CSE-K-VI-Sem) Roll No: 2218018 (18)

CSE-K-VI Sem

Session: 2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

**GRAPHIC ERA HILL UNIVERSITY, DEHRADUN**

**CERTIFICATE**

This is to certify that Mr. Brahm Sagar Gaur (Roll No: 2218018 (18)) has developed a mini project on “YouTube Video Summarizer” for the BTech CSE VI Semester Mini Project Lab in Graphic Era Hill University, Dehradun. The project carried out by students is their own work as best of my knowledge.

Date: 20/07/24

Ms. Himadri Vaidya

Class Co-Ordinator

CSE-K-VI-Sem

(CSE Dept.)

GEHU DEHRADUN

**ACKNOWLEDGEMENT**

We would like to thank our project coordinator Ms. Himadri Vaidya and her encouragement throughout this project and for having faith in us.

We want to thank our parents for their continuing support and encouragement. We also want to thank them for providing us with the opportunity to reach this far in our studies.

At last but not least, we are greatly indebted to all other people who directly or indirectly helped us during the work.

**INDEX**

1. **Introduction**

* Challenges of Online Video Content
* Solution: YouTube Video Summarizer Application

1. **Functionalities**

* User Input and Link Capture
* Transcript Extraction
* Summary Generation
* Summary Display

1. **Benefits**

* Increased Efficiency for Research and Information Seeking
* Improved Accessibility for Text-Preferring Users and People with Visual Impairments
* Enhanced Content Discovery through Efficient Video Evaluation

1. **Technical Specifications**

* Frontend: Streamlit (User Interface Creation)
* Backend: Google GenerativeAI Service (including Gemini-Pro model) and youtube\_transcript\_api library

1. **Code Explanation**

* Library Imports
* API Key Configuration (Note: Secured in Separate File)
* Prompt Definition
* extract\_transcript\_details Function
* generate\_gemini\_content Function

1. **Implementation**

* Transcript Extraction Process
* Summary Generation Process
* Streamlit Integration (User Interface and Interaction)

1. **Further Considerations**

* Error Handling Enhancements
* User Experience Improvements (Original Transcript Display, Summary Length Control, Navigation)
* Advanced Summarization Techniques (Other Models, Multilingual Support)

1. **Conclusion**

* Recap and Key Achievements
* Future Directions (Advanced Techniques, Multilingual Support, Speaker Identification, Sentiment Analysis)
* Overall Impact: Empowering Users

**INTRODUCTION**

The vast amount of video content available online can be overwhelming. Viewers often struggle to identify relevant videos or lack the time to watch them entirely. This report introduces a YouTube video summarizer application that addresses this challenge. By leveraging the power of Google's GenerativeAI service and the user-friendly interface provided by Streamlit, the application allows users to efficiently grasp the key points of YouTube videos without the need to watch them in their entirety.

1. **Difficulty Identifying Relevant Videos:** With so many videos, users often struggle to find content relevant to their interests or needs. The sheer volume of videos can make it time-consuming to locate the ones that provide the information they seek.
2. **Time Commitment of Watching Videos:** Even after finding a potentially relevant video, users might not have the time to watch it entirely. This is especially true if they are only looking for specific information within the video.

The YouTube video summarizer application acts as a solution to these challenges. It emphasizes that the application allows users to:

* **Gain Insights Quickly:** By generating summaries of YouTube videos, the application empowers users to grasp the key points of the content without having to invest the time in watching the entire video. This is particularly beneficial for viewers seeking specific details within a video.
* **Improved Accessibility:** The application caters to users who prefer text summaries over video content. This can be particularly valuable for individuals with visual impairments or those who struggle with processing information presented through audio-visual mediums.

**FUNCTIONALITIES**

The core functionality of the application revolves around summarizing YouTube videos. Here is a breakdown of the user experience:

1. **User Input:** This section describes how users interact with the application. It explains that users are presented with a designated field within the Streamlit interface. This field allows them to enter a YouTube video link by simply pasting the URL.
2. **Link Capture:** Once a user enters a YouTube video link, the application utilizes Streamlit's capabilities to capture the provided URL. This captured link is then used for further processing within the application.
3. **Transcript Extraction:** This section dives into the process of retrieving the video's transcript. The report explains that the application leverages an external library called youtube\_transcript\_api. This library automates the process of fetching the transcript text associated with the captured YouTube video link. The captured video ID (extracted from the URL) is used by the library to access the transcript data from YouTube or a relevant database.
4. **Summary Generation:** Here, the report explains how the application generates a concise summary of the video content. It highlights two key aspects:
   * **Prompt:** A pre-defined prompt is used to instruct the AI model (specifically the Gemini-Pro model from Google's Generative AI service) on how to process the transcript text. This prompt specifies the desired outcome, which is to summarize the key points of the transcript.
   * **Model Interaction:** The captured transcript text, along with the defined prompt, is sent to the Gemini-Pro model. This powerful AI model then analyzes the transcript based on the prompt's instructions and generates a summary that highlights the video's main points.
5. **Summary Display:** Finally, the functionalities page explains how the application presents the generated summary to the user. It details that the summary text received from the Gemini-Pro model is displayed within the Streamlit interface. This allows users to view the concise overview of the video content directly within the application window.

The application takes a YouTube video link as input, retrieves the transcript, utilizes an AI model for summarization, and presents a user-friendly summary within the Streamlit interface.

**BENIFITS**

This application offers several benefits to users:

**Increased Efficiency:** This section emphasizes how the application saves the user's valuable time. By generating summaries of video content, users can quickly grasp the main points without having to watch the entire video. This is particularly beneficial for situations where users are:

* + **Researching a Topic:** When exploring a topic and encountering numerous videos, the application allows users to efficiently determine if a video aligns with their needs by reviewing the summary before committing to watching it entirely.
  + **Seeking Specific Information:** If users are looking for specific details within a video, the summary can guide them directly to the relevant sections, saving them time spent navigating through the entire video.

**Improved Accessibility:** This section explains how the application caters to a wider audience by providing an alternative way to access video content. It highlights the advantages for users who:

* + **Prefer Text Summaries:** Some users might find it easier or more convenient to consume information through text rather than video. The application provides summaries that condense the video's key points into a readable format.
  + **Have Visual Impairments:** Users with visual impairments might struggle to watch videos entirely. The application offers text summaries as an alternative way to access the video's content.

**Enhanced Content Discovery:** This section explores how the application can improve the process of finding relevant video content. By offering summaries, the application allows users to:

* + **Quickly Evaluate Videos:** Summaries provide a quick overview of the video's content, enabling users to determine if a video aligns with their interests before investing time in watching it. This can streamline the content discovery process and help users find the most relevant videos for their needs.
  + **Identify Diverse Content:** The application can be used to explore a wider range of videos by allowing users to efficiently assess the content before committing to watching them. This can potentially expose users to new perspectives and information sources.

**TECHNICAL SPECIFICATIONS**

The application leverages two key technologies:

1. Frontend: Streamlit

This section explains that the application utilizes Streamlit, a Python library specifically designed for creating web applications. Streamlit offers several advantages that make it ideal for this application:

* + **Simplified User Interface Creation:** Streamlit provides a collection of functions that allow developers to build user interfaces with minimal coding. This reduces development time and effort compared to building interfaces from scratch.
  + **User-Friendly Interaction Elements:** Streamlit offers various pre-built elements like text input fields, buttons, and markdown formatting. These elements enable the creation of an interactive and user-friendly interface for the YouTube video summarizer.
  + **Rapid Prototyping and Deployment:** Streamlit allows for rapid development and deployment of web applications. This makes it suitable for creating a proof-of-concept application like the video summarizer and facilitates easy deployment for wider user access.

1. Backend: Google Generative AI Service (including Gemini-Pro model) and youtube\_transcript\_api library

This section delves into the functionalities powering the core video summarization process:

* + **Google GenerativeAI Service:** The report highlights that the application leverages Google's GenerativeAI service, a platform providing access to powerful AI models capable of understanding and manipulating human language. This service offers various functionalities, but the application specifically utilizes:
    - **Large Language Models (LLMs):** These are advanced AI models trained on massive amounts of text data. They can generate human-quality text, translate languages, write various kinds of creative content, and answer your questions in an informative way. The application leverages this capability for summarizing video content.
  + **Gemini-Pro Model:** The report specifies that within the GenerativeAI service, the application utilizes the Gemini-Pro model. This is a specific LLM within the service known for its advanced capabilities in text summarization and content generation. By using Gemini-Pro, the application aims to generate concise and informative summaries of the video transcripts.
  + **youtube\_transcript\_api library:** This section explains that the application integrates an external library called youtube\_transcript\_api. This library plays a crucial role by automating the process of:
  + **Transcript Extraction:** The library interacts with Youtube's API or relevant databases to retrieve the transcript text associated with the provided YouTube video link. This eliminates the need for manual transcript retrieval, streamlining the process.

**CODE EXPLAINATION**

## **Code Example Breakdown with Specifications**

The provided code example demonstrates the core functionalities of the YouTube video summarizer application. Here is a detailed breakdown of each section with specifications:

**1. Library Imports:**

Python

import streamlit as st  
from dotenv import load\_dotenv  
import os  
import google.generativeai as genai  
from youtube\_transcript\_api import YouTubeTranscriptApi

* streamlit as st**:** Imports the Streamlit library and assigns it the alias st. This allows for concise use of Streamlit functions throughout the code.
* from dotenv import load\_dotenv**:** Imports the load\_dotenv function from the dotenv library. This function is used to load environment variables from a .env file (explained later).
* import os**:** Imports the os library, which provides various functionalities including environment variable access (potentially used for functionalities not shown in the provided code).
* import google.generativeai as genai**:** Imports the generativeai library from Google's GenerativeAI service. This library provides access to various functionalities including interacting with LLMs.
* from youtube\_transcript\_api import YouTubeTranscriptApi**:** Imports the YouTubeTranscriptApi class from the youtube\_transcript\_api library. This class offers functionalities for retrieving transcripts from YouTube videos.

**2. API Key Configuration (Not shown in provided code, but important for implementation):**

The code utilizes a .env file to securely store the Google API key required for accessing the GenerativeAI service. This is not explicitly shown in the provided code, but it is a crucial step for real-world implementation. Here is how it might be achieved:

1. Create a file named .env in the same directory as your Python script.
2. Add a line to the .env file with the following format: GOOGLE\_API\_KEY=your\_api\_key\_here. Replace your\_api\_key\_here with your actual Google API key obtained from the Google Cloud Platform console.
3. Use the load\_dotenv() function (already imported) to load these environment variables before initializing the GenerativeAI service.

**3. Prompt Definition:**

Python

prompt="""You are YouTube video summarizer. You will be taking the transcript text  
and summarizing the entire video and providing the important summary in points  
with detailed description. Please provide the summary of the text given here: """

This section defines a variable named prompt that holds the text instruction provided to the Gemini-Pro model. This prompt specifies the desired outcome, which is to summarize the video based on the provided transcript text. The prompt also requests a summary with "important points with detailed description," indicating a preference for informative summaries.

**4.** extract\_transcript\_details **Function:**

Python

def extract\_transcript\_details(youtube\_video\_url):  
 try:  
 video\_id=youtube\_video\_url.split("=")[1]  
   
 transcript\_text=YouTubeTranscriptApi.get\_transcript(video\_id)  
  
 transcript = ""  
 for i in transcript\_text:  
 transcript += " " + i["text"]  
  
 return transcript  
  
 except Exception as e:  
 raise e

This function takes a YouTube video URL as input and performs the following actions:

* **Video ID Extraction (Specification: String Splitting):** It splits the URL string at the = symbol and retrieves the second element, which is typically the video ID within a YouTube URL.
* **Transcript Retrieval (Specification:** youtube\_transcript\_api **library):** It utilizes the YouTubeTranscriptApi.get\_transcript(video\_id) function to retrieve the transcript text associated with the extracted video ID. This interacts with Youtube's API or relevant databases.
* **Transcript Concatenation:** The function iterates through the retrieved transcript data (potentially a list of transcript segments) and combines them into a single string variable named transcript.
* **Error Handling (Specification: Exception Handling):** The code includes a try-except block to handle potential exceptions during transcript retrieval. If an exception occurs (e), it is re-raised using raise e.

**5.** generate\_gemini\_content **Function:**

Python

def generate\_gemini\_content(transcript\_text,prompt):  
  
 model=genai.GenerativeModel("gemini-pro")  
 response=model.generate\_content(prompt+transcript\_text)  
 return response.text

This function takes the extracted transcript text and the defined prompt as input and performs the following actions:

* **Generative Model Initialization (Specification:** genai.GenerativeModel**):** It creates a new instance of the GenerativeModel class from the genai library. This class provides functionalities for interacting with various LLMs within the GenerativeAI service. The argument "gemini-pro" specifies that we want to use the Gemini-Pro model for this task.
* **Content Generation (Specification:** model.generate\_content**):** It utilizes the generate\_content method of the created model instance (model). This method takes two arguments:
  + prompt: The previously defined prompt instructing the model on how to process the transcript text (summarize the video with important points and detailed descriptions).
  + transcript\_text: The extracted transcript text retrieved from the YouTube video. By calling model.generate\_content with these arguments, the function sends the transcript and instructions to the Gemini-Pro model for processing.
* **Response Handling (Specification: Accessing Text Output):** The generate\_content method returns a response object containing the generated summary text. The function then accesses the text content of this response using the .text attribute and returns it.
* This function acts as an intermediary between the application and the Gemini-Pro model. It prepares the transcript and prompt, interacts with the model to generate the summary, and retrieves the generated text for further processing within the application.

**IMPLEMENTATION**

The implementation page of the report dives deeper into the inner workings of the YouTube video summarizer application, building upon the functionalities explained earlier. Here is a breakdown of the key aspects:

**Transcript Extraction:**

* **Video ID Extraction:** When a user enters a YouTube video link, the application utilizes Streamlit functions to capture the provided URL. This captured URL is then passed to the extract\_transcript\_details function.
* extract\_transcript\_details **Function Breakdown:**
  + This function extracts the video ID from the YouTube link. It achieves this by splitting the URL string at the = symbol and retrieving the second element, which typically represents the video ID within a YouTube URL.
  + It then leverages the YouTubeTranscriptApi.get\_transcript(video\_id) function to retrieve the transcript text associated with the extracted video ID. This interacts with Youtube's API or relevant databases to fetch the transcript data.
  + The retrieved transcript data might be a list containing segments of the video transcript. The function iterates through this data and combines all the transcript segments into a single string variable named transcript. This combined string represents the entire video transcript.
  + The function incorporates error handling using a try-except block. If any exceptions occur during transcript retrieval (e), the error is re-raised using raise e. This ensures the application gracefully handles potential issues with transcript retrieval.

**Summary Generation:**

* generate\_gemini\_content **Function Breakdown:**
  + This function takes the extracted transcript text and the defined prompt as input.
  + It creates an instance of the GenerativeModel class from the genai library, specifying "gemini-pro" to utilize the Gemini-Pro model for summarization.
  + The function then uses the generate\_content method of the created model instance. This method sends the transcript text and the defined prompt (instructing the model to summarize) to the Gemini-Pro model.
  + The generate\_content method returns a response object containing the generated summary text. The function retrieves this text using the .text attribute and returns it.

**Streamlit Integration:**

* **User Interface and Interaction:** Streamlit functions are used to create the user interface within the application. This includes elements like:
  + A title ("YouTube Transcript to Detailed Notes Converter") displayed at the beginning.
  + A text input field where users can enter the YouTube video link.
  + A button labeled "Get Detailed Notes" that triggers the summarization process.
* **Link Capture and Thumbnail Display (Not explicitly shown in provided code):** Upon entering a link, the application extracts the video ID and utilizes YouTube APIs (not included in the provided code) to retrieve a thumbnail image associated with the video. This thumbnail image is then displayed within the interface.
* **Summary Display:** Clicking the "Get Detailed Notes" button triggers the following actions:
  + The extract\_transcript\_details function is called to retrieve the transcript text from the provided YouTube video link.
  + If the transcript retrieval is successful, the generate\_gemini\_content function is called with the extracted transcript and the defined prompt.
  + The returned summary text is then displayed within the Streamlit interface using markdown formatting ("## Detailed Notes:") followed by the actual summary content.

The implementation of the application bridges the gap between the user interaction facilitated by Streamlit and the core functionalities handled by the extract\_transcript\_details and generate\_gemini\_content functions. It outlines how the user-provided link is processed to extract the transcript, how the transcript and prompt are used to generate a summary with the Gemini-Pro model, and finally, how the generated summary is presented to the user within the application's interface.

**FURTHER CONSIDERATIONS**

**1. Error Handling:**

The report emphasizes the importance of robust error handling mechanisms. While the provided code incorporates a basic try-except block for transcript retrieval, additional considerations could be:

* **Handling Various Error Scenarios:** The application could be extended to handle specific error types related to API calls, model failures, or invalid user input (e.g., incorrect YouTube link format).
* **Informative User Messages:** Upon encountering errors, the application should display informative messages to guide users. These messages could explain the issue, suggest troubleshooting steps, or provide alternative video links.
* **Retry Mechanisms:** Consider implementing functionalities to retry failed API calls or suggest alternative video links in case of persistent errors.

**2. User Experience Improvements:**

The report explores ways to enhance the user experience of the application:

* **Displaying Original Transcript:** An option could be added to display the original transcript text alongside the generated summary. This allows users to refer to specific portions of the video if needed. This functionality might require additional UI elements within Streamlit to allow collapsible/expandable sections for the transcript.
* **Summary Length Control:** The report suggests allowing users to adjust the desired summary length based on their preference. This could involve providing options within the interface or potentially exploring functionalities within the Gemini-Pro model itself to control the conciseness of the generated summary.
* **Navigation to Specific Timestamps:** The report proposes integrating functionalities to navigate to specific timestamps within the YouTube video based on the summarized points. This would require further processing of the transcript to identify key points and potentially utilizing YouTube APIs to link timestamps with identified points within the video. This is an advanced feature that could significantly enhance the application's value.

**3. Advanced Summarization Techniques:**

The report acknowledges the potential for exploring more advanced summarization techniques:

* **Integrating Other Summarization Models:** Consider incorporating other summarization models from Google's GenerativeAI service or exploring different AI-powered summarization techniques beyond Gemini-Pro. This could involve comparing the outputs of different models or potentially using ensemble approaches that combine multiple models for improved accuracy and comprehensiveness.
* **Multilingual Support:** The application could be extended to support videos in multiple languages by integrating functionalities for multilingual transcript extraction and summarization. This could leverage multilingual capabilities offered by the GenerativeAI service or by integrating other translation APIs.

**CONCLUSION**

This report has explored the YouTube video summarizer application, a tool designed to address the challenges users face in the ever-growing landscape of online video content. Identifying relevant videos and efficiently grasping their key points can be time-consuming, especially with the sheer volume of available content.

**Recap and Key Achievements**

This application leverages the power of AI-powered summarization to offer users a solution. By integrating Streamlit for a user-friendly interface and Google's GenerativeAI service (specifically the Gemini-Pro model) for video summarization, the application provides concise summaries that highlight the key points of videos. This report has detailed the application's functionalities, technical specifications, and implementation process.

**Future Directions: Expanding the Capabilities**

The exciting future directions for this application lie in further exploration of advanced video summarization techniques. Here are some potential areas of development:

* **Advanced Summarization Techniques:** While the application currently utilizes the Gemini-Pro model, future iterations could explore integrating other summarization models or combining approaches (extractive and abstractive) to potentially improve the accuracy and comprehensiveness of the generated summaries.
* **Multilingual Support:** Expanding the application's capabilities to handle videos in multiple languages can significantly broaden its reach. This could involve leveraging the multilingual functionalities offered by the GenerativeAI service or by integrating other translation APIs.
* **Speaker Identification and Summarization:** By incorporating functionalities to identify different speakers within a video, the application could potentially generate summaries specific to each speaker's contribution. This would enhance the user experience, especially for videos with multiple viewpoints or presenters.
* **Sentiment Analysis Integration:** Integrating functionalities to analyze the sentiment or tone of the video content alongside summarizing key points provides a more comprehensive understanding of the video. This could offer valuable insights into the overall message or mood conveyed within the video.

**Overall Impact: Empowering Users**

By incorporating these potential advancements, the YouTube video summarizer application can evolve into a powerful tool for users navigating online video content. It can empower users to:

* **Efficiently Find Relevant Videos:** Quickly assessing video content through summaries allows users to identify videos that align with their interests and needs, saving valuable time in the searching process.
* **Grasp Key Points Quickly:** Summaries offer users a concise overview of the video's content, enabling them to grasp the essential information without having to watch the entire video.
* **Access Video Content in Preferred Formats:** For users who prefer text summaries over video content, the application provides an alternative way to access and understand video information.