

YouTube Transcript, Summarizer and Keyword Extractor

A project submitted to the Bharathidasan University
in partial fulfillment of the requirements
for the award of the Degree of

MASTER OF SCIENCE IN DATA SCIENCE

Submitted by

Pavithra S
225229125

Under the guidance of

Dr. R. Mallikka M. Sc., M. Phil., M. Tech., Ph. D.
Assistant Professor



Handwritten signature

DEPARTMENT OF DATA SCIENCE BISHOP HEBER COLLEGE (AUTONOMOUS)

"Nationally Reaccredited at A++ Grade with a CGPA of 3.69 out of 4 in NAAC IV Cycle"
(Recognized by UGC as "College of Excellence")
(Affiliated to Bharathidasan University)

TIRUCHIRAPPALLI-620 017

MARCH – 2024

DECLARATION

I hereby declare that the project work presented is originally done by me under the guidance of **Dr. R. Mallikka M. Sc., M. Phil., M. Tech., Ph. D., Assistant Professor, Department of Data Science, Bishop Heber College (Autonomous), Tiruchirappalli-620 017** and has not been included in any other thesis/project submitted for any other degree.

Name of the Candidate : Pavithra S

Register Number : 225229125

Batch : 2022-2024



Signature of the Candidate

S. Pavithra (225229125)


Dr. R. Mallikka M. Sc., M. Phil., M. Tech., Ph. D.,
Assistant Professor,
Department of Data Science,
Bishop Heber College (Autonomous),
Tiruchirappalli – 620017

Date: 20/03/24

CERTIFICATE

This is to certify that the project work entitled **"YouTube Transcript, Summarizer and Keyword Extractor"** is a bonafide record work done by **S.Pavithra(Reg.No:225229125)** in partial fulfillment of the requirements for the award of the degree of **MASTER OF SCIENCE IN DATA SCIENCE** during the period **2023 - 2024.**

Place: Tiruchirappalli


Signature of the Guide



**DEPARTMENT OF DATA SCIENCE
BISHOP HEBER COLLEGE (AUTONOMOUS),**

"Nationally Reaccredited at A++ Grade with a CGPA of 3.69 out of 4 in NAAC IV Cycle"

(Recognized by UGC as "College of Excellence")

(Affiliated to Bharathidasan University)

TIRUCHIRAPPALLI - 620017

Date: 20/03/24

Course Title: Project

Course Code: P22DS4PJ

CERTIFICATE

The Viva-Voce examination for the candidate **S. Pavithra**(Reg No: 225229125) was held on 22/03/2024

Signature of the Guide

Signature of the HOD

Examiners:

1. A.M. 22/03/24

2. M.L. 22/3/24

ACKNOWLEDGEMENT

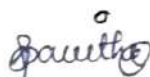
I thank **GOD ALMIGHTY** for having given me the strength and patience. the knowledge to do my project work successfully.

As a gift from the heaven. I had my **Parents** in my life who had been with me in every step of my winning.

I sincerely thanks to **Dr. J. Princy Merlin, M.Sc., SET., B.Ed., M.Phil., Ph.D., PGDCI.,** Principal. Bishop Heber College (Autonomous), Trichy-17, for granting permission and necessary facilities to do the project work very effectively.

My sincere thanks to **Dr. R. JEMIMA PRIYADARSHINI, MCA., M.Phil., Ph.D.,** Coordinator and Associate Professor Department of Data Science, Bishop Heber College (Autonomous), Tiruchirappalli-17, for her motivation and inspire suggestions to bring this project, a great success.

I extent my gratitude to my guide **Dr. R. Mallikka M. Sc., M. Phil., M. Tech., Ph. D., Assistant Professor,** Department of Data Science, Bishop Heber College (Autonomous). Trichy-17. for her guidance and support.



Signature of the candidate

S. Pavithra (Reg No: 225229125)

225229125

ORIGINALITY REPORT

3%

SIMILARITY INDEX

3%

INTERNET SOURCES

2%

PUBLICATIONS

1%

STUDENT PAPERS

PRIMARY SOURCES

1

ijeecs.iaescore.com

Internet Source

2%

2

www.ijraset.com

Internet Source

1%

3

www.atlantis-press.com

Internet Source

<1%

4

ijarcce.com

Internet Source

<1%

5

Submitted to National College of Ireland

Student Paper

<1%

6

Submitted to Universidade Estadual de
Campinas

Student Paper

<1%

7

Submitted to University of Carthage

Student Paper

<1%

8

qubixity.net

Internet Source

<1%

9

www.irjmets.com

Internet Source

<1%

ABSTRACT

The "YTV Summarizer" application is designed to simplify the management and comprehension of YouTube video content. It offers a straightforward solution for users to effectively handle and extract key insights from the vast array of videos available on the platform. Utilizing tkinter, the application provides a user-friendly interface, making it easy for individuals to transcribe, summarize, and identify important information within videos. Users simply input a YouTube video ID, initiating the process of retrieving the video's transcript through the YouTube Transcript API. This transcript serves as the foundation for the application's functionality. Leveraging the capabilities of the sumy library, the application employs Latent Semantic Analysis (LSA) to distill the content of the transcript into a concise summary. By condensing complex information into easily understandable points, the summarization process enhances the accessibility and comprehension of the video's content. Additionally, the yake library is integrated into the application to extract relevant keywords from the transcript. These keywords provide valuable insights into the main topics and themes discussed in the video, facilitating a deeper understanding of its content. The combination of transcript summarization and keyword extraction empowers users across various domains, including students, researchers, educators, and content creators, to efficiently grasp the essence of video content. Whether for academic study, content creation, or personal enrichment, the "YTV Summarizer" application simplifies the consumption and analysis of YouTube videos. With its intuitive interface and robust functionality, the application enables users to effortlessly extract actionable insights, thereby enhancing their overall experience in navigating and leveraging online video content.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
1	Introduction 1.1 Motivation 1.2 Existing System 1.3 Proposed System	1
2	Literature Review	8
3	Logical Development 3.1 Architecture System 3.2 DFDs	10
4	Database Design 4.1 Table Design 4.2 Data Dictionary 4.3 Relational Diagram	14
5	Program Design 5.1 Technologies Used 5.2 Algorithms Applied 5.3 Method Of Summarization 5.4 Accessing YouTube Video and Selecting a Video	17
6	User Interface Design And Evaluation 6.1 User Interface Design 6.2 Test Graphical User Interface 6.3 Summary Metrics Analysis	33
7	Conclusion 7.1 Summary 7.2 Limitations	37

	7.3 Future Works	
	References	39
	APPENDIX – A: SOURCE	41
	CODE	50
	APPENDIX – B: OUTPUT	
	SCREEENSLOTS	

CHAPTER-01

INTRODUCTION

The "YTV Summarizer " application is a versatile tool designed to streamline the process of transcribing, summarizing, and extracting essential information from YouTube videos. Its primary objective is to provide users with a convenient and efficient means of handling video content, offering a range of features tailored to simplify these tasks. Developed using Python's Tkinter library, the application boasts a user-friendly interface that ensures ease of navigation and accessibility for individuals seeking to effectively manage their video materials.

Upon launching the application, users are greeted with a straightforward interface designed for intuitive interaction. The title "YTV Summarizer" succinctly encapsulates the primary function of the app: delivering a concise yet comprehensive summary of video content. Complementing this is the accompanying subtitle, "Transcribe, Summarize, and Save The Essence.," which serves to clarify the key functionalities offered by the application, setting clear expectations for users.

The initial step in utilizing the application involves inputting the YouTube video ID of the desired content. This crucial step ensures that the software accurately retrieves the specified video for processing. Once the ID is entered, users can initiate the transcription process by selecting the "Get Transcript" button, prompting the application to fetch the video's transcript using the YouTube Transcript API.

Behind the scenes, the application leverages advanced techniques to enhance the user experience beyond simple transcription. It incorporates a summarization feature driven by Natural Language Processing (NLP) capabilities. By integrating the Sumy library, the application employs algorithms to identify key sentences from the transcript and generate a concise summary of the video content. Additionally, it utilizes the YAKE algorithm to extract important keywords, enriching the summary with relevant terms and concepts.

The summarization process seamlessly integrates into the application's interface, providing users with a convenient platform to access the summarized content. The summary is presented in a dedicated text box, accompanied by the extracted keywords displayed in a separate section. This visual representation enables users to quickly grasp the main points of the video and understand its key themes.

Furthermore, the application offers functionality for users to save both transcripts and summaries for future reference. This feature enables users to export the content as PDF files, facilitating easy storage and sharing of information. This ensures that users can access the transcribed and summarized content at their convenience, enhancing the utility of the application for various purposes, including research, education, and content creation.

In addition to its robust features, the application prioritizes usability to ensure a seamless user experience. A "Clear" button is readily available to quickly reset the interface, allowing users to switch between tasks effortlessly. Overall, the "Rapid Recap" application empowers users to efficiently transcribe, summarize, and extract key insights from YouTube videos. Its combination of user-friendly design and advanced NLP capabilities makes it a valuable tool for individuals across diverse domains, facilitating the rapid comprehension and analysis of video content.

1.1 Motivation:

In our modern world overflowing with information, YouTube videos can be a treasure trove of knowledge. However, with so many videos out there, it can be hard to find the time to watch them all, let alone understand the key points. This is where the "YTV Summarizer" app comes in as your trusty sidekick!

Imagine a helpful friend who watches YouTube videos for you and then tells you the important stuff. That's essentially what the YTV Summarizer does. It's like having a super-powered assistant who can understand the spoken words in a video and then write them down for you. This written version of the video, called a transcript, is like having the video's script right at your fingertips.

But wait, there's more! The YTV Summarizer doesn't just give you the whole transcript – it also knows how to find the most important parts. Imagine skimming a long article to find the main points – that's what the app does with the transcript. It uses clever tools like "Sumy" to automatically shorten the transcript and highlight the key ideas, like the main points of a story or the important details in a lecture.

And to make things even easier, the app can also identify the most important words in the video, like keywords in a search engine. Think of these keywords like little flags sticking out of the transcript, pointing you to the main topics being discussed. So, whether you're a student

trying to understand a complex topic, a researcher gathering information, or simply someone curious about a new subject, the YTV Summarizer can help you quickly grasp the essence of the video.

Using the app is as easy as pie! It has a friendly interface with clear buttons and a simple design, so anyone can use it without any hassle. Think of it like a well-organized desk for your video summaries – you can easily find what you need when you need it.

But the fun doesn't stop there! The app lets you save these summaries and transcripts as PDF files, like digital notebooks for your YouTube learnings. Imagine having a library of all the important information you've gleaned from videos, all neatly organized and ready for you to revisit whenever you need a refresher.

So, if you're feeling overwhelmed by the vast amount of video content online, don't worry! The YTV Summarizer is here to help you make sense of it all. It's a convenient tool that simplifies the process of extracting valuable insights from YouTube videos, making it a perfect companion for anyone who wants to learn and explore the endless world of online information.

1.2 Existing System:

The current frameworks for summarizing YouTube videos utilize a combination of cutting-edge algorithms to automatically extract key information from video content. These algorithms are derived from various fields such as machine learning, natural language processing, computer vision, and graph-based techniques. Machine learning algorithms enable the system to identify patterns and extract relevant features from the video, while natural language processing techniques analyze textual data associated with the video to provide context. Computer vision methods analyze visual content to identify important scenes or moments, and graph-based algorithms model the relationships between different elements of the video to emphasize significant content.

Simultaneously, the graphical user interface (GUI) of these systems plays a crucial role in facilitating user interaction and control over the summarization process. It typically offers user-friendly controls for video playback, allowing users to navigate through the content and review it comprehensively. Moreover, the GUI provides options for users to personalize the summarization, such as selecting different algorithms or adjusting parameters based on their

preferences. Visualization tools like timelines and keyframe thumbnails assist users in understanding the structure and context of the video, while textual summaries are displayed to convey the essence of the content effectively.

In essence, the GUI acts as a bridge between users and the sophisticated algorithms operating in the background, delivering a seamless and user-friendly experience. It empowers users to efficiently generate concise summaries of YouTube videos for various purposes, whether for research, entertainment, or information gathering. By combining advanced algorithms with an intuitive interface, these systems aim to enhance the accessibility and utility of video content on platforms like YouTube.

1.3 Proposed System:

The proposed system is a sophisticated application created to make transcribing and summarizing audio or video content easier, particularly focusing on YouTube videos. This system meets the growing need for tools that can process content efficiently. It aims to offer users a simple and intuitive interface for extracting useful information from multimedia sources rapidly and efficiently.

The transcription module in the proposed system allows users to input a YouTube video ID and effortlessly retrieve its transcript using the YouTubeTranscriptApi. It efficiently extracts spoken content from the video and presents it in an easy-to-read format. This functionality enhances accessibility to video content by providing accurate transcripts, benefiting users in research, content creation, or educational endeavors.

The summarization module utilizes advanced natural language processing techniques to condense lengthy transcripts into concise summaries. Through the Latent Semantic Analysis (LSA) algorithm, it identifies key themes and extracts essential information from the transcript. This feature saves time and enhances comprehension by providing users with comprehensive overviews of the video content, particularly beneficial for extracting insights or summaries from extensive video content.

The system incorporates a keyword extraction feature using the YAKE algorithm. This functionality enables users to identify significant terms and phrases within the transcript, offering insights into the core topics discussed in the video. By extracting keywords, users can quickly understand the main focus of the video content and effectively categorize it. This

capability enhances content understanding and organization, facilitating easier navigation and utilization of the extracted information.

The user interface of the proposed system is designed to be intuitive and user-friendly, developed using the Tkinter library. This ensures accessibility and ease of use for all users. It includes input fields for entering YouTube video IDs, buttons for starting transcription and summarization processes, and text boxes for displaying transcripts, summaries, and extracted keywords. The design emphasizes clarity and simplicity, allowing users to navigate the application effortlessly.

After transcribing and summarizing tasks are completed, the system enables users to save the generated content as PDF documents. Users have the option to save the transcript, summary, or both, making it convenient to access and share the processed information. Additionally, the system may include a database feature for storing metadata associated with processed videos, aiding users in tracking and managing their transcription and summarization activities effectively.

The proposed system is designed to be adaptable and scalable, accommodating future enhancements and integrations with external services or APIs. It can seamlessly integrate with cloud storage platforms, enabling users to store and access processed documents flexibly and conveniently. Furthermore, the system architecture allows for the addition of language support and advanced algorithms to improve transcription accuracy and summarization quality.

The System offers a comprehensive solution for transcribing and summarizing YouTube videos, meeting the diverse needs of users seeking efficient content processing tools. With its advanced functionalities, user-friendly interface, and focus on integration and scalability, the system aims to streamline content analysis and facilitate knowledge discovery across various domains.

1.3.1 Hardware Specification:

The proposed system's hardware specifications are designed to efficiently handle the various processing tasks involved in transcribing and summarizing YouTube videos. A multi-core CPU with a clock speed of at least 2.0 GHz is recommended to effectively manage the computational demands, allowing for parallel execution and faster performance. For memory, a minimum of 4 GB of RAM is advised, with 8 GB or more recommended for smoother operation, especially with larger files or multiple processes running simultaneously. In terms of storage, a minimum of 100 GB of disk space is suggested to accommodate application files, temporary data, and downloaded video transcripts, with an SSD being preferred for improved responsiveness and reduced loading times.

While a dedicated graphics card isn't essential for basic functionality, it can enhance performance for tasks like video playback and graphical rendering. However, integrated graphics processors commonly found in CPUs suffice for general usage, with the option for users to upgrade to a discrete GPU for enhanced performance if needed. These hardware specifications strike a balance between performance, affordability, and scalability, ensuring users experience smooth operation and efficient handling of transcription and summarization tasks.

1.3.2 Software Specifications:

The software is designed to be highly versatile, compatible across a range of operating systems including Windows, macOS, and various Linux distributions such as Ubuntu and Fedora. Its foundation lies in Python, a language known for its simplicity and extensive library support, particularly in natural language processing (NLP). Leveraging Python ensures efficient implementation of complex functionalities like text transcription, summarization, and keyword extraction, with ongoing support and updates from the developer community.

For the graphical user interface (GUI), the software utilizes Tkinter, known for its simplicity and ease of use, ensuring an intuitive user experience. To retrieve transcripts from YouTube videos, the software employs the YouTubeTranscriptApi library, facilitating seamless interaction with YouTube's transcript data. Text summarization is achieved through the Sumy library, offering various algorithms including Latent Semantic Analysis (LSA) to condense lengthy transcripts into concise summaries. Keyword extraction from transcripts is

facilitated by the YAKE library, providing insights into core topics discussed in the content. Finally, the software enables users to save transcripts, summaries, and extracted keywords as PDF documents using the ReportLab library, enhancing convenience and portability. Overall, these carefully selected tools and libraries ensure a seamless experience for transcribing, summarizing, and analysing YouTube videos across diverse platforms, enhancing accessibility and usability in content processing tasks.

CHAPTER-02

LITERATURE REVIEW

The research paper titled[1] "Summary and Keyword Extraction from YouTube Video Transcript" by Shraddha Yadav, Arun Kumar Behra, Chandra Shekhar Sahu, and Nilmani Chandrakar presents an innovative Apache to video transcript summarization utilizing Natural Language processing (NLP) techniques. By employing both extractive and abstractive summarization methods, the researchers aim to provide users with concise summaries and significant keywords extracted from video transcripts. This approach offers a time-saving solution, enabling users to efficiently access vital information without the need to watch the entire video, thereby reducing effort and enhancing Productivity.

In contrast, the work by Anika Dilawari and Muhammad Usman Ghani Khan, titled[2] "Abstractive Summarization of Video Sequences," focuses on abstractive summarization techniques using deep neural network models such as RCNN. While their approach emphasizes the succinctness of the summary, it overlooks considerations such as time restrictions and memory efficiency, potentially limiting its practical applicability in real-world scenarios.

The review paper titled[3] "Review of Automatic Text Summarization Techniques & Methods" by Adhika Pramita, Supriadi Rustad, Abdul Shukur, and Affandy provides a comprehensive overview of automatic text summarization techniques, highlighting the strengths and weaknesses of various approaches. Despite employing systematic review techniques, the paper identifies limitations in fuzzy-based approaches, particularly in addressing semantic issues, and emphasizes the need to bridge gaps in extractive summarization methods.

Similarly, the survey conducted by Ishitva Awasthi, Kuntal Gupta, Prajbot Singh Bhojal, Anand, and Piyush Kumar, titled[4] "Natural Language processing (NLP) based Text Summarization - A Survey," explores both extractive and abstractive summarization methods. While acknowledging the benefits of analyzing linguistic and statistical features for summarization, the survey notes the challenge of determining the most effective technique due to the variability in specific use cases.

Furthermore, the research paper authored by Parth Rajesh Dedhia, Hardik Pradeep, and Meghana Naik, titled [5]"Research on Abstractive Text Summarization Methods," investigates the effectiveness of seq2seq models and Encoder-Decoder mechanisms for abstractive

summarization. However, the study identifies limitations in handling multiple documents simultaneously, posing challenges in scalability and efficiency.

In contrast, the research paper referenced in [6] poses an Automatic NLP-based LSA summarization algorithm that operates on video subtitles, utilizing Latent Semantic Analysis (LSA) to extract features from sentences. This approach offers computational efficiency and requires no training data, making it suitable for real-time summarization of video content.

Moreover, the paper mentioned in [7] presents the "YouTube Transcript Summarizer," which utilizes the Hugging Face Transformer for abstractive summarization of YouTube transcripts. By leveraging RESTAPI in the backend, the model generates condensed summaries from input transcripts, offering a streamlined solution for summarizing video content.

The article discussed in [8] highlights the evolution of video summarization and skimming techniques, emphasizing their importance in video management systems. The authors' work on movie skimming, incorporating rhythmic analysis of audio and visual content, demonstrates advancements in summarization methods tailored for specific video types.

Kiyomarsi [9] compares different automatic text summarization methods that use fuzzy with the summarization made by humans and concludes that there is no significant difference between them, and the fuzzy method developed is more economical. Sah et al. [10] also use video summarization with two tasks: first, dividing the video into parts as joint photographic experts group (JPEG) files and then arranging it. Then they record the parts of the result according to word frequency analysis of speech transcripts. Several elements may be clearly examined for YouTube videos: visual pictures, video metadata, (such as time and creator), music, and transcripts.

One approach to defining video content is by transcribing the videos, relying on video transcripts opens it up to computational textual analysis techniques to look for word interrelationships, often used words or sentences, and themes [11].

In existing research explores various approaches to text and video summarization, including extractive and abstractive methods, our proposed model aims to enhance efficiency and accessibility by providing multilingual summarization capabilities. By leveraging NLP techniques and considering the diverse needs of users, our model strives to offer a comprehensive solution for summarizing video content effectively across different languages.

CHAPTER-03

LOGICAL DEVELOPMENT

3.1 Architecture Design:

The architectural design of the YTV Summarizer is meticulously crafted with a user-centric prioritizing simplicity, accessibility, and efficiency at every stage of its workflow. This design philosophy aims to cater to users of varying technical abilities, ensuring that they can navigate the application effortlessly and accomplish their tasks with minimal friction.

Integration with YouTube Transcript API:

The integration with the YouTube Transcript API is crucial to our software's architecture. It enables easy retrieval of transcripts from specified YouTube videos, connecting our software seamlessly with YouTube's vast multimedia content. Behind the scenes, a robust backend system efficiently manages API requests and data, acting as the intermediary between our frontend and the YouTube Transcript API. When a user requests a transcript through our interface, the backend system communicates with the API, retrieves the transcript data, and presents it to the user. This process shields users from technical complexities, ensuring a smooth experience. Ultimately, this integration forms the foundation of our software, empowering users to access and analyze multimedia content effortlessly.

Text Processing Phase:

The text processing stage is crucial for refining transcript data retrieved from the YouTube Transcript API before analysis. It involves removing timestamps, speaker IDs, and formatting artifacts while standardizing spelling and punctuation for coherence. Noise like background disturbances and filler words are filtered out to ensure accurate analysis and summarization. This phase ensures subsequent stages yield meaningful insights reflecting the original content's essence.

Keyword Extraction:

Keyword extraction is essential for summarizing videos in our application. It identifies key terms and phrases to encapsulate core themes. Through advanced algorithms and linguistic analysis, relevant keywords are discerned, ensuring a comprehensive summary. By leveraging frequency, context, and semantic relevance, the application identifies main topics efficiently.

Sophisticated linguistic and statistical analysis methods, including natural language processing (NLP) and semantic analysis, are employed for comprehensive transcript analysis. This enhances the summarization process, providing users with concise yet meaningful summaries. Keyword extraction categorizes content and enriches user understanding, playing a crucial role in video summarization.

Summarization Algorithm:

At the heart of the application's functionality is its summarization algorithm. The application's summarization algorithm utilizes advanced natural language processing techniques to condense transcripts into concise summaries, integrating extracted keywords for emphasis. It empowers users to swiftly comprehend video content by discerning key information and highlighting significant concepts. This algorithm distills transcripts into essential components, ensuring that resulting summaries encapsulate main themes effectively. Overall, the algorithm enhances users' ability to grasp core video content efficiently, facilitating a streamlined and insightful summarization process.

Database Integration and Export Options:

The application integrates with a SQLite database for storing transcripts and summaries, enabling convenient management and access over time. Users can efficiently organize their data and export transcripts as PDF files for offline access and sharing. This combination enhances accessibility and usability, providing seamless access to stored content.

The architectural design of the YTV Summarizer effectively combines user-centric design principles with a robust backend functionality. By prioritizing simplicity, accessibility, and efficiency, the application videos a seamless and efficient user experience, enabling users to effortlessly transcribe, summarize, and manage YouTube videos. This user-centric apace ensures that the application remains accessible and intuitive, catering to the diverse needs of its user base.

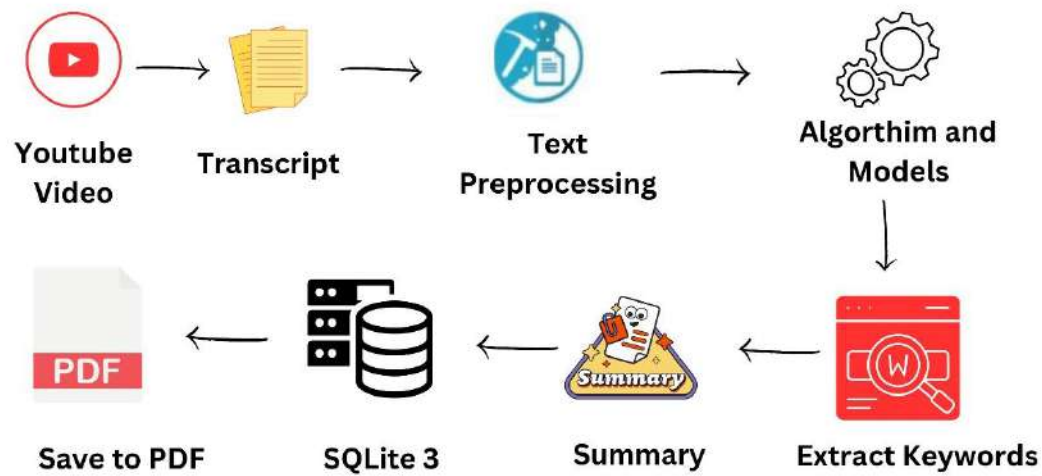


Figure 3.1

3.2 DFDs:

The YouTube Summary application operates within a clear and intuitive user-centric flow, ensuring straightforward access to the transcription and summarization functionalities. At the core of its functionality lies a sequence of processes initiated by user input in the form of a YouTube video ID. This input serves as the catalyst for a series of actions orchestrated seamlessly within the application.

The first step in the process involves fetching the transcript from YouTube, a task executed by the "Get Transcript" process. Leveraging the YouTube API, this process communicates with external systems to retrieve the transcript data associated with the provided video ID. This interaction ensures that users gain access to the raw textual content of the video, laying the groundwork for further analysis and summarization.

Once the transcript is obtained, it undergoes processing within the "Generate Summary" process. Here, the Sumy library comes into play, applying advanced text summarization algorithms to distill the lengthy transcript into a concise summary of the video content. This step condenses the transcript's key points and themes, providing users with a quick overview of the video's content and facilitating efficient information extraction.

Furthermore, the application incorporates a SQLite database to facilitate data persistence and long-term record-keeping. The "Save to Database" process takes the video details, including the YouTube ID, link, transcript, and summary, and stores them securely within the database. This functionality not only enables users to retrieve and review past transcriptions but also contributes to data integrity by ensuring that valuable information is preserved for future reference.



Figure 3.2

CHAPTER-04

DATABASE DESIGN

4.1 Table Design:

The SQLite database table designed for the YouTube transcriber application is structured to efficiently store and manage essential information related to YouTube videos. The table, named `videos`, is meticulously organized to facilitate easy retrieval and manipulation of data. Here's a detailed explanation of each field in the table:

ID: This field serves as a primary key and ensures the uniqueness of each record in the table. It is automatically incremented for every new entry, guaranteeing that each video in the database has a distinct identifier. The ID field is essential for referencing and linking other information related to a specific video.

YouTube ID: The `youtube_id` field stores the unique identifier associated with a YouTube video. This identifier is extracted from the video link and serves as a reference point for linking to the original content on the YouTube platform. It enables easy retrieval of the video's metadata and facilitates interactions with the YouTube API.

YouTube Link: The `youtube_link` field contains the complete YouTube video link, providing users with a direct URL to access the video on the YouTube platform. This link is constructed by appending the YouTube ID to the standard video URL, ensuring seamless navigation to the original content.

Transcript: The `transcript` field is designed to store the textual content of the video's transcript. This information is obtained using the YouTube Transcript API and is stored as plain text within the database. Storing the transcript allows users to access and review the spoken content of the video, facilitating further analysis.

Summary: The `summary` field is used to store a summarized version of the video's transcript. This summary is generated using the Latent Semantic Analysis (LSA) summarization technique, condensing the key points and main ideas of the video into a concise format. Storing the summary enables users to quickly grasp the essence of the video's content without having to review the entire transcript.

Timestamp: The `timestamp` field automatically records the date and time when a record is inserted into the database. This field provides a chronological reference for tracking when video data was added or modified. It allows users to monitor the history of video uploads and summarization activities, facilitating audit trails and data management.

By structuring the data into this table format, the YouTube transcriber application can efficiently manage and retrieve information about YouTube videos. Users can transcribe, summarize, and save the essence of videos while maintaining a comprehensive record of these activities in the database. This organized Apache enhances data accessibility, facilitates analysis, and supports the overall functionality of the application.

4.2 Data Dictionary:

It allows users to transcribe YouTube videos, summarize the transcripts, and save both the transcript and summary to a PDF file. Additionally, it stores the video details, transcript, and summary in a SQLite database.

Field	Data Type	Description
id	INTEGER	Unique identifier for each record. Automatically incremented for new entries.
youtube_id	TEXT	Unique identifier associated with a YouTube video. Obtained from the video link.
youtube_link	TEXT	Complete YouTube video link, created by appending the YouTube ID to the standard video URL.
transcript	TEXT	Textual content of the video's transcript, obtained using the YouTube Transcript API.
summary	TEXT	Summarized version of the video's transcript, generated using Latent Semantic Analysis (LSA).
timestamp	DATETIME	Date and time when the record was inserted into the database.

Table 4.2

4.3 Relational Diagram:

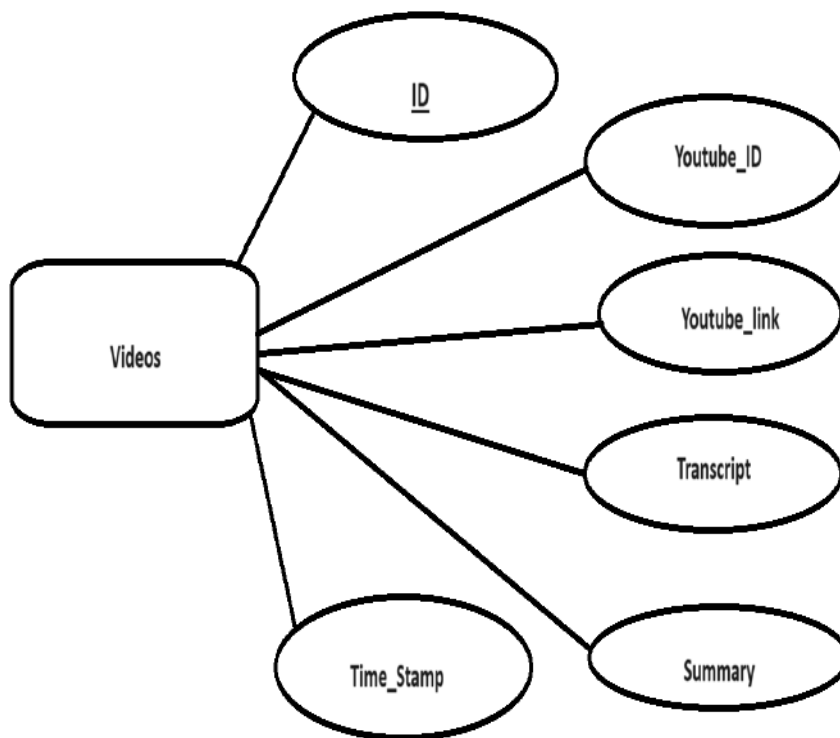


Figure 4.3

CHAPTER-05

PROGRAM DESIGN

The software utility, named the YouTube Video Transcription and Summarization Tool, has been meticulously crafted to streamline the often intricate of transcribing and summarizing YouTube videos. Its primary objective is to offer users a straightforward method to extract pertinent information from video content efficiently. This document meticulously elucidates the design of the tool, viding a comprehensive overview of each segment.

Phase 1: Importing Libraries and Setting Up Environment:

At the outset of developing the YouTube Video Transcription and Summarization Tool, the focal point lies in establishing a conducive development environment by importing requisite libraries and configuring indispensable tools. This phase lays the foundation for subsequent developmental stages and ensures that developers are equipped with indispensable resources to construct the application efficiently.

In order to craft user-friendly graphical interfaces for the application, Tkinter, a standard Python library for GUI development, is imported. This integration empowers developers to fashion interfaces that facilitate seamless interaction with the tool's functionalities. Tkinter furnishes an array of widgets and layout options, thereby enabling developers to tailor the appearance and behavior of the application's interface to cater to the specific needs of users.

The incorporation of YouTubeTranscriptApi enables the tool to grammatically fetch video transcripts based on their unique IDs. This library assumes a pivotal role in accessing transcript data from YouTube videos, thereby enabling users to transcribe video content seamlessly within the application. By integrating YouTubeTranscriptApi, developers can automate the transcript retrieval obviating the need for manual transcription and conserving users' valuable time and effort.

ReportLab is introduced to facilitate dynamic PDF generation, thereby enabling users to store transcripts in a structured and portable format. Leveraging ReportLab, developers can grammatically generate PDF files containing transcript data, thereby enhancing the usability

and shareability of transcribed content. This functionality empowers users to conveniently archive and share transcripts, ensuring that the information remains accessible and well-organized.

`yake` is a Python library specialized in keyword extraction from text, employing a blend of statistical and linguistic methodologies. Through its algorithms, it discerns key phrases or terms that encapsulate the primary topics or themes within a given textual input. This process involves analyzing the text's structure and content, leveraging statistical insights to identify significant patterns, frequencies, and correlations among words. What distinguishes `yake` is its adaptability, offering users a customizable approach to keyword extraction by allowing adjustments to parameters like keyword length, weighting schemes, and language-specific settings, thereby enhancing the accuracy and relevance of extracted keywords. The utility of these keywords spans various applications such as document categorization, indexing, and search engine optimization (SEO). In document categorization, they aid in organizing textual data based on underlying topics, while in indexing systems, they serve as valuable metadata for efficient search and retrieval. Moreover, in SEO, keywords play a pivotal role in enhancing the visibility and ranking of web pages by aligning their content with user search queries. In essence, `yake` provides a robust and efficient solution for uncovering crucial insights and themes embedded within textual content, making it an invaluable tool for text mining and natural language processing endeavors.

Sumy, an indispensable library, is harnessed for the purpose of text summarization. This library offers a plethora of algorithms that condense lengthy text documents into concise summaries. By employing Sumy, developers can generate summaries of video transcripts, thereby enabling users to quickly glean important insights without perusing the entire transcript. The integration of Sumy augments the efficiency of content analysis and empowers users to extract valuable information from video content effectively.

SQLite3 is incorporated for database operations, enabling efficient storage and retrieval of video information such as YouTube IDs, links, transcripts, summaries, and timestamps. This integration ensures persistent data storage within the application, thereby facilitating seamless access to video data and enhancing overall functionality. Through the utilization of SQLite3, developers can implement robust data management capabilities, enabling users to efficiently organize and manage their video content.

To ensure accurate tracking of video accessing times, the datetime module is imported for timestamp manipulation. By leveraging the datetime module, developers can record the time when videos are transcribed and summarized, thereby enhancing the precision and reliability of the application's accessing capabilities. This ensures that users have access to up-to-date information and enables developers to effectively monitor and optimize the performance of the application.

The inclusion of these indispensable libraries and tools lays the groundwork for the subsequent developmental stages, facilitating the seamless implementation of key functionalities within the YouTube Video Transcription and Summarization Tool. This phase ensures that developers are equipped with the requisite resources to construct a robust and efficient application that effectively caters to the needs of users.

Phase 2: The Creation of the Application Architecture

During the second phase of development for the YouTube Video Transcription and Summarization Tool, the focus lies on crafting a robust application architecture to underpin its functionality. At the heart of this architecture stands the `YoutubeTranscriberApp` class, which serves as the linchpin of the system, responsible for orchestrating its essential functions and coordinating the interplay between various components to ensure optimal performance.

At its core, the `YoutubeTranscriberApp` class leverages Tkinter, a Python library renowned for its ability to create graphical user interfaces (GUIs), to set up the GUI components. Through Tkinter's feature-rich toolkit, the class meticulously designs and constructs an intuitive user interface, encompassing an array of elements including labels, entry fields, buttons, and text boxes. These meticulously crafted GUI components provide users with an interactive platform, facilitating seamless navigation and operation of the application.

The `YoutubeTranscriberApp` class takes on the responsibility of integrating with an SQLite database, thereby establishing a structured repository for storing and managing video data. By creating requisite tables and defining the database schema, the class lays the groundwork for efficient storage and retrieval of critical information such as YouTube IDs, links, transcripts, summaries, and timestamps. This seamless integration with SQLite not only ensures data persistence but also upholds data integrity within the application, fostering seamless access to video data.

Furthermore, the class is tasked with creating GUI widgets tailored for user interaction. These widgets encompass a spectrum of elements ranging from labels conveying title and author information to entry fields facilitating input of YouTube video IDs, alongside buttons for triggering actions such as transcript retrieval and summary generation, and text boxes for displaying transcript and summary content. By encapsulating these GUI elements within the ``YoutubeTranscriberApp`` class, the application fosters modularity, extensibility, and maintainability, adhering to the principles of object-oriented design.

In essence, by consolidating the application's functionality within the ``YoutubeTranscriberApp`` class and adhering to object-oriented design principles, the system establishes a structured and organized architecture. This architectural blueprint champions code reusability, scalability, and ease of maintenance, empowering developers to efficiently manage and enhance the YouTube Video Transcription and Summarization Tool over time.

The application architecture embraces key object-oriented design principles such as encapsulation, inheritance, and polymorphism, which serve to delineate clear boundaries between different concerns and modularize functionality. This design ethos augments code readability, flexibility, and ease of testing, thereby facilitating effective collaboration among developers and iterative refinement of the application's development process.

The architectural design of the YouTube Video Transcription and Summarization Tool embodies a structured and organized approach, with the ``YoutubeTranscriberApp`` class standing as the central pillar orchestrating various functionalities. This design ethos underscores scalability, maintainability, and extensibility, enabling the application to evolve and adapt to evolving requirements and user needs over time.

Phase 3: Creating the Graphical User Interface (GUI):

In the meticulous crafting of the graphical user interface (GUI) for the YouTube Video Transcription and Summarization Tool, every aspect is carefully considered to ensure a seamless and intuitive user experience. Tkinter, a powerful GUI toolkit for Python, serves as the foundation for creating an interface that not only meets the functional requirements of the application but also delights users with its ease of use and practicality.

Labels, strategically positioned throughout the interface, serve as signposts that guide users through the application's functionalities. By prominently displaying metadata such as the application's title and author, these labels provide users with essential context, fostering a sense of familiarity and clarity as they navigate through the tool. This attention to detail ensures that users can easily orient themselves within the application and understand its purpose and functionality from the outset.

Entry fields are seamlessly integrated into the interface to streamline user input, particularly for entering YouTube video IDs. These entry fields are designed to be intuitive and user-friendly, allowing users to input the unique identifiers of the videos they wish to transcribe and summarize with minimal effort. By simplifying the data entry process, entry fields enhance user engagement and participation, empowering users to access and interact with video content effortlessly.

Buttons, serving as interactive controls, play a pivotal role in enabling users to perform various actions within the application. Whether it's initiating transcript retrieval, generating summaries, saving transcripts as PDF files, or clearing data, buttons provide users with clear and intuitive cues for interacting with the tool. Through their well-defined actions and visual cues, buttons empower users to navigate the application with confidence, ensuring that essential tasks can be executed efficiently and effectively.

Text boxes, strategically placed within the interface, serve as containers for displaying both transcript and summary content. These text boxes are designed to present information in a clear, organized, and visually appealing manner, enabling users to review and analyze video content seamlessly. By providing a visual representation of the transcribed and summarized information within the application itself, text boxes facilitate user comprehension and decision-making, enhancing the overall usability and utility of the tool.

In essence, the GUI of the YouTube Video Transcription and Summarization Tool is meticulously crafted to prioritize usability, functionality, and user satisfaction. By leveraging Tkinter's capabilities and incorporating elements such as labels, entry fields, buttons, and text boxes, the interface offers users an intuitive and immersive experience. This thoughtful design ensures that users can navigate the application effortlessly, interact with video content seamlessly, and accomplish their tasks with ease, ultimately enhancing their overall satisfaction and engagement with the tool.

Phase 4: Implementing YouTube Transcription :

In the YouTube Video Transcription and Summarization Tool, the implementation of YouTube transcription is meticulously crafted to ensure a seamless and user-centric experience. The ``get_transcription`` method plays a pivotal role in orchestrating this process, encapsulating the various steps involved in fetching, parsing, and presenting the video transcript data.

When invoked, the ``get_transcription`` method initiates communication with the YouTube Transcript API, leveraging its functionalities to retrieve transcript content associated with the specified video ID. This communication is conducted with precision, considering factors such as video availability, permissions, and format to ensure the accurate retrieval of transcript data. By establishing a robust connection with the API, the tool guarantees reliable access to the textual representation of the video content.

Following the retrieval of transcript data, the method undertakes a comprehensive parsing and processing phase to prepare the data for presentation within the GUI. Each segment of the transcript undergoes meticulous analysis, with a focus on extracting text while maintaining the integrity of the original content. This parsing process is essential for ensuring that users receive a coherent and accurate representation of the video content in textual format, laying the groundwork for further interaction and analysis.

Subsequently, the processed transcript data is seamlessly integrated into the GUI, enabling users to access and interact with the text of the video in real-time. The GUI is thoughtfully designed to prioritize user-friendliness and intuitiveness, with clear navigation elements and visual cues facilitating seamless exploration of the transcript content. Users can effortlessly scroll through the transcript, search for specific keywords, or jump to specific sections, enhancing their ability to engage with the textual representation of the video.

In addition to providing the raw transcript, the ``get_transcription`` method enhances the user experience by generating a summary of the transcript content. Leveraging advanced text summarization algorithms offered by the Sumy library, this feature condenses the transcript into a concise overview of the video's main points and themes.

This summary serves as a valuable resource for users seeking quick insights into the video content, enabling them to efficiently grasp important information without the need to read through the entire transcript.

Furthermore, the tool ensures seamless integration between the transcript retrieval process and other functionalities, such as keyword extraction and PDF generation. By synchronizing these features within a cohesive framework, the tool enhances user productivity and efficiency, enabling them to seamlessly transition between different tasks while analyzing video content.

Overall, the meticulous implementation of YouTube transcription within the YouTube Video Transcription and Summarization Tool underscores the program's commitment to providing a comprehensive and user-centered solution for video analysis and understanding. Through careful integration of API communication, data processing, and GUI presentation, the tool empowers users to interact with video content efficiently and extract meaningful insights effortlessly, enhancing their overall experience and satisfaction with the tool.

Phase 5: Summarization and Keyword Extraction

In the phase of summarization and keyword extraction within the YouTube Video Transcription and Summarization Tool, the application employs advanced text processing techniques to distill key insights from the video transcript. This phase is crucial as it enhances the utility of the tool by providing users with concise summaries of video content and relevant keywords that encapsulate its main themes.

The ``generate_summary`` method plays a central role in this phase, utilizing the Sumy library to generate a summary of the transcript. Initially, the transcript data is parsed and tokenized using the `PlaintextParser` and `Tokenizer` classes, respectively, to prepare it for summarization. Then, the `LsaSummarizer` is employed to perform Latent Semantic Analysis (LSA) on the parsed transcript, extracting essential sentences that encapsulate the main points of the video. By condensing the transcript into a succinct summary, users can quickly grasp the core content of the video without the need to review the entire transcript.

Additionally, the tool leverages the YAKE library to extract keywords from the transcript, providing users with insight into the main topics and themes discussed in the video. The ``extract_keywords`` method of the `KeywordExtractor` class is utilized to analyze the

transcript text and identify key phrases that represent significant concepts. These keywords are extracted based on statistical and linguistic analysis, ensuring that they accurately reflect the content of the video. By presenting these keywords alongside the summary, users can gain a comprehensive understanding of the video's content at a glance.

Furthermore, the extracted keywords are appended to the end of the summary, enhancing the user's ability to quickly identify the main topics discussed in the video. This integration of summarization and keyword extraction streamlines the process of analyzing video content, allowing users to extract meaningful insights efficiently. Overall, this phase of the application's development underscores its commitment to providing users with a robust tool for video transcription, summarization, and keyword extraction, empowering them to engage with video content in a more insightful and efficient manner.

```
def generate_summary(self, transcript):
    parser = PlaintextParser.from_string(transcript, Tokenizer("english"))
    summarizer = LsaSummarizer()
    summary_sentences = summarizer(parser.document, 3)
    summary = ' '.join(str(sentence) for sentence in summary_sentences)

    # Extract keywords using YAKE
    extractor = KeywordExtractor()
    keywords = extractor.extract_keywords(text=transcript)
    keywords_text = '\n'.join([keyword for keyword, _ in keywords])
```

Phase 6: Generating PDF Transcripts

In this phase of development, the YouTube Video Transcription and Summarization Tool enhances its functionality by incorporating the capability to generate PDF documents. This addition greatly enhances the utility of the tool, allowing users to conveniently save transcripts and summaries for future reference. At the core of this functionality lies the `save_as_pdf` method, which seamlessly integrates with the ReportLab library to facilitate the dynamic creation of PDF documents based on the transcript and summary data extracted within the application.

When users trigger the `save_as_pdf` method, the program leverages the robust features of the ReportLab library to initiate the creation of a PDF document. This involves setting up a canvas and defining the layout and formatting parameters for the document, ensuring that the content is presented in an organized and visually appealing manner. The transcript and

summary text, obtained from the graphical user interface (GUI), are meticulously formatted and inserted into the PDF document to ensure readability and accessibility.

Customization plays a vital role in optimizing the presentation of the transcript and summary text within the PDF file. Parameters such as font styles, sizes, line spacing, and margins are carefully tailored to enhance the overall legibility and usability of the document. By adhering to established principles of document design and typography, the PDF transcripts generated by the tool offer users a cohesive and professional viewing experience, ensuring that important information is conveyed effectively.

Furthermore, the incorporation of PDF generation into the YouTube Video Transcription and Summarization Tool amplifies its usefulness by providing users with a versatile and portable format for storing transcript and summary data. As PDF files are utilized for storage, important metadata such as YouTube IDs, links, and timestamps are simultaneously recorded in the SQLite database. This seamless integration enables users to effortlessly retrieve and manage transcript and summary content within the application, promoting a smooth and accessible experience.

The meticulous implementation of PDF technology within the tool underscores its commitment to delivering a comprehensive and user-centric solution for video analysis and documentation. Through seamless integration with the ReportLab library and robust database operations, the tool empowers users to efficiently store and access both transcript and summary content in a structured and organized manner, facilitating enhanced productivity and knowledge management. Overall, the addition of PDF generation capability enhances the utility and versatility of the YouTube Video Transcription and Summarization Tool, ensuring that users can effectively capture and utilize valuable insights from video content.

Phase 7: Managing Databases and Persisting Data

The YouTube Video Transcription and Summarization Tool focuses on database management and data persistence, utilizing SQLite as its primary database management system. The implementation of database operations is crucial for effectively organizing and managing video-related information within the application.

Upon initialization of the application, a SQLite database is created to establish a structured framework for storing and retrieving data. Within this database schema, a dedicated

table is defined to store essential information such as YouTube IDs, links, transcripts, summaries, and timestamps. This structured approach ensures efficient organization and retrieval of data, facilitating seamless data management.

Database operations, including data insertion and retrieval, are seamlessly integrated into the application's functionality. As users transcribe and summarize YouTube videos, relevant video data is automatically inserted into the SQLite database, ensuring the persistence and integrity of the data. SQL queries are utilized to interact with the database, executing commands to insert new data or retrieve existing data as required. By adhering to best practices for database management, such as employing parameterized queries and handling transactions, the program ensures the security and reliability of database operations.

To address potential issues during database operations, robust error handling mechanisms are implemented. Exception handling strategies are employed to gracefully manage errors that may occur, such as database connection failures or data integrity violations. This proactive approach to error handling enhances the stability and reliability of database interactions, ensuring uninterrupted functionality for users.

Efficient management of connections is essential for optimizing database operations. The application utilizes connection pooling techniques to effectively manage database connections, reducing resource usage and enhancing performance. This ensures that database connections are established and released in a controlled manner, preventing resource depletion and improving scalability.

Overall, the meticulous execution of database operations and data persistence underscores the program's commitment to providing a reliable and user-friendly solution. By leveraging SQLite for efficient data storage and retrieval, adhering to best practices for database management, and implementing robust error handling and connection management strategies, the program ensures seamless data management and integrity. This enables users to confidently transcribe, summarize, and manage YouTube videos, promoting a smooth and efficient user experience.

5.1 Technologies Used:

Tkinter:

- Tkinter is a standard GUI (Graphical User Interface) library in Python.
- It provides various widgets and tools for creating graphical interfaces, such as buttons, labels, entry fields, and more.
- Tkinter is widely used due to its simplicity and ease of integration with Python applications.
- In the provided code, Tkinter is utilized to create the user interface for the YouTube transcriber application. It's responsible for creating windows, labels, entry fields, buttons, etc., enabling users to interact with the application.

youtube_transcript_api:

- The YouTube Transcript API is a library that allows developers to retrieve the transcripts of YouTube videos programmatically.
- It provides methods to fetch the transcript data based on the unique video ID of a YouTube video.
- This API simplifies the process of accessing transcript data, which can then be used for various purposes such as analysis, summarization, or display.
- In the provided code, `youtube_transcript_api` is used to fetch the transcript of a YouTube video based on the user-provided video ID.

Reportlab:

- ReportLab is a Python library used for creating and manipulating PDF documents.
- It provides tools for generating PDF files from scratch, as well as modifying existing PDF documents.
- ReportLab supports various features such as text formatting, image embedding, and page layout customization.

- In the provided code, ``reportlab`` is employed to save the transcript and summaries generated by the application as PDF files. This allows users to store the extracted information in a widely-used format for future reference or sharing.

Sumy:

- Sumy is a Python library for automatic text summarization.
- It offers several algorithms for summarizing text, including LSA (Latent Semantic Analysis), Luhn, LexRank, and more.
- Sumy simplifies the task of generating concise summaries from large bodies of text, which can be useful for tasks such as information retrieval or content analysis.
- In the provided code, ``sumy`` with the LSA summarization algorithm is utilized to generate summaries of YouTube video transcripts, providing users with condensed versions of the content for easier understanding or reference.

Yake:

- YAKE (Yet Another Keyword Extractor) is a Python library used for extracting keywords from text.
- It employs algorithms to identify significant terms or phrases within a document that represent its key topics or themes.
- Keyword extraction can aid in tasks such as document indexing, topic modeling, or content categorization.
- In the provided code, ``yake`` is used to extract keywords from the YouTube video transcripts. These keywords help users identify the main themes or topics discussed in the video content, providing additional insights beyond the summarized text.

NLTK (Natural Language Toolkit) :

- NLTK is a Python library widely used in natural language processing (NLP) tasks.
- It provides various tools and resources for tasks such as tokenization, stemming, lemmatization, part-of-speech tagging, parsing, and more.
- In the provided code, NLTK is used for tokenization, specifically to tokenize the transcript obtained from the YouTube video. The ``word_tokenize`` function from

NLTK is utilized for this purpose, ensuring that the summarization algorithm operates on individual words rather than entire sentences.

Overall, these technologies collectively enable the YouTube transcriber application to fetch video transcripts, summarize content, extract keywords, and save the results in a convenient format for users to access and utilize. Each technology plays a crucial role in different aspects of the application's functionality, contributing to its overall effectiveness and utility.

5.2 Algorithm Used:

Latent Semantic Analysis (LSA) is a statistical method utilized in Natural language processing and statistics retrieval to find the underlying shape of textual content. It ambitions to seize the latent (hidden) semantic relationships among phrases and files in a corpus. Here's a proof of the way LSA works:

Term-Document Matrix: LSA starts via building a matrix illustration of the textual content corpus, referred to as the term-document matrix. Each row of the matrix corresponds to a time period (word), and each column corresponds to a document (textual content). The matrix factors constitute the frequency of prevalence of every term in every report.

Dimensionality Reduction: LSA then applies a mathematical approach referred to as Singular Value Decomposition (SVD) to reduce the dimensionality of the time period-record matrix. SVD decomposes the matrix into 3 separate matrices: U , Σ , and V . By keeping most effectively the most sizable dimensions (singular values), LSA reduces noise and captures the crucial semantic relationships.

Semantic Space: The decreased-dimensional area received from SVD is referred to as the semantic space. In this area, files and phrases are represented as vectors, wherein every measurement corresponds to a latent semantic concept. LSA effectively clusters together phrases and files that have similar semantic meanings.

Cosine Similarity: To determine the similarity between files or phrases, LSA computes the cosine similarity among their corresponding vectors in the semantic area. Documents with comparable content material can have vectors that align intently, resulting in a higher cosine similarity score.

Summarization: In the context of text summarization, LSA identifies the most essential sentences in a record with the aid of considering their semantic relevance to the overall content material. It does this by choosing sentences which can be closest in semantic area to the centroid of all sentences, efficiently capturing the essence of the textual content in a concise summary.

5.3 Method of Summarization:

The `generate_summary` method in the provided code is responsible for generating a summary of the transcript. It utilizes the `PlaintextParser` from the `sumy.parsers.plaintext` module to parse the transcript text, then utilizes the `LsaSummarizer` from the `sumy.summarizers.lsa` module to summarize the parsed text. Additionally, it utilizes the YAKE library to extract keywords from the transcript.

Parsing the Transcript: The transcript text is parsed using the `PlaintextParser.from_string()` method, which takes the transcript text and an instance of a tokenizer as arguments. In this case, it uses the English tokenizer.

Summarizing the Transcript: The parsed transcript is then summarized using the `LsaSummarizer`, which summarizes the text into a specified number of sentences. In this case, it summarizes the text into three sentences.

Extracting Keywords: Keywords are extracted from the transcript using the YAKE library. The `extract_keywords()` method is used to extract keywords from the transcript text.

Appending Keywords to Summary: The keywords extracted are appended to the end of the summary text.

5.4 Accessing YouTube and Selecting a Video:

Users manually navigate to the YouTube website and select a video for transcription. With advancements in voice recognition technology, automatic transcription from YouTube has become reliable, often providing transcriptions shortly after video uploads. This process involves users actively browsing the YouTube platform to find videos relevant to their transcription needs. Once a suitable video is identified, users can initiate transcription processes, leveraging the platform's built-in transcription capabilities or external tools for

further transcription accuracy or customization. The accessibility and reliability of automatic transcription offered by YouTube significantly streamline the transcription process, facilitating efficient extraction of spoken content for various purposes such as research, analysis, or content creation.

Getting YouTube Video URL:

Users input the video's unique ID into our app, initiating a request to the YouTube API for the video content. The API acts as a mediator, enabling communication between our app and YouTube's servers, resulting in the retrieval of the video content for subsequent transcription and analysis.

NLP Processing and Transcription:

Tokenization breaks down the transcript into smaller units after extraction, aiding in analysis by segmenting the text. Text cleaning enhances transcript readability and consistency by removing unnecessary characters and standardizing text format, ensuring more accurate and reliable analysis results.

Summarizing the Transcript:

The application utilizes the Latent Semantic Analysis (LSA) algorithm to summarize the transcript. LSA represents textual data in a high-dimensional vector space and identifies semantic structures through techniques like Singular Value Decomposition (SVD). It assigns weights to sentences, selecting the most salient ones to construct a concise summary, facilitating efficient analysis and understanding of the video content.

Extracting Keywords:

The YAKE algorithm comprehensively analyzes the transcribed text to extract keywords. It identifies key terms or phrases encapsulating primary themes and topics discussed in the video, evaluating frequency, context, and significance to generate a concise set of metadata. These extracted keywords aid in understanding the video's core themes and facilitate efficient navigation and retrieval of relevant information, enriching the overall user experience.

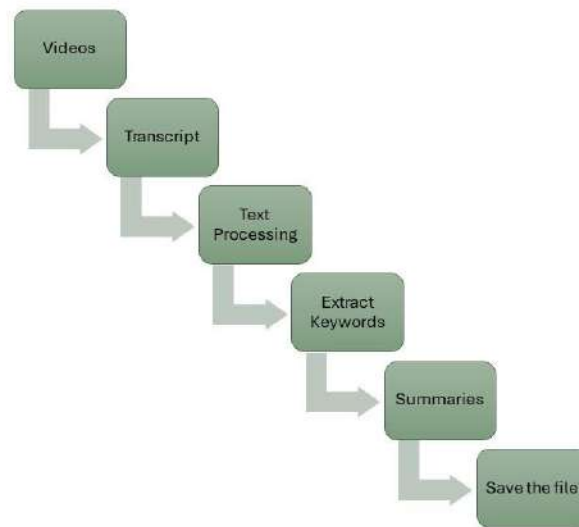


Figure 5.4

Through the systematic execution of the aforementioned methodology, our "Rapid Recap" application empowers users to efficiently transcribe, summarize, and extract key insights from YouTube videos. By integrating advanced NLP techniques and algorithms, we facilitate the rapid comprehension and analysis of multimedia content, enhancing productivity and knowledge acquisition for our users.

CHAPTER-06

USER INTERFACE DESIGN AND EVALUATION

Designing and evaluating the graphical user interface (GUI) of YTV Summarizer is crucial to ensure a user-friendly and intuitive experience for users interacting with the application. This process involves creating an interface that simplifies the interaction with the underlying functionality of the system, while also considering aspects such as usability, accessibility, and aesthetics.

6.1 User Interface Design:

The user interface (UI) design of the application, "Rapid Recap," prioritizes simplicity, functionality, and aesthetic appeal to ensure a seamless experience for users seeking to transcribe and summarize YouTube videos. At the forefront of the UI stands a bold and inviting title, "Rapid Recap," conveying the application's purpose of offering swift video summaries. Beneath this title, users encounter a well-structured input section where they are prompted to input the YouTube ID of the desired video. Clear labels and entry fields guide users through this process, ensuring clarity and ease of use.

Once users input the YouTube ID and initiate the transcription process, the interface dynamically displays the transcribed content in a scrolled text area. This feature allows users to effortlessly navigate through the transcript, regardless of its length, facilitating a comprehensive understanding of the video's content. Below the transcript display lies a dedicated section for the summarized version of the video content. This succinct summary enables users to quickly grasp the key points of the video, enhancing efficiency and usability.

To empower users with essential functionalities, the UI incorporates intuitive action buttons strategically placed for easy access. These buttons, including "Get Transcript," "Save as PDF," and "Clear," provide users with the necessary tools to transcribe, save, and manage their summarized content seamlessly. Each button is adorned with clear labels and positioned thoughtfully to optimize user interaction and workflow.

Complementing the functional aspects of the UI is a visually appealing background image that adds a touch of elegance and charm. This background image enhances the overall ambiance of the application, contributing to a pleasant and engaging user experience. Together, these design elements culminate in a user interface that embodies simplicity, efficiency, and aesthetic sophistication, elevating the user experience of transcribing and summarizing YouTube videos.

6.2 Summary Metrics Analysis:

The summary generated by the YouTube Transcriber application provides a condensed version of the video transcript, aiming to capture the essential information presented in the video. To assess the effectiveness of the summary, several metrics can be applied:

Readability Analysis:

Assessing text readability is crucial for ensuring accessibility to the intended audience. Two widely used metrics for this purpose are the Flesch Reading Ease Score and the Flesch-Kincaid Grade Level. The former quantifies readability based on sentence length and syllables per word, with higher scores indicating easier comprehension. Conversely, the latter estimates the educational grade level required for understanding, considering factors like sentence length and word complexity. By utilizing these metrics, we can evaluate the clarity and accessibility of a summary and adjust enhance comprehension as necessary. Higher Flesch Reading Ease Scores and lower Flesch-Kincaid Grade Levels typically signify improved accessibility, indicating that the summary is more easily comprehensible to its target audience. Analyzing these metrics enables effective evaluation and refinement of the summary's clarity and accessibility.

Coherence Evaluation:

Assessing coherence in a summary involves analyzing the flow of information and logical connections between sentences to ensure clarity and cohesion. One aspect of coherence evaluation entails examining the presence of clear topic sentences, which serve as guiding statements for each paragraph or section, summarizing its main idea. Additionally, coherence can be enhanced by assessing the density of transition words, such as "however," "therefore," and "consequently," which facilitate smooth transitions between ideas and improve the overall coherence of the text. By scrutinizing these elements, coherence metrics provide valuable insights into the structural integrity of the summary, enabling refinement to enhance clarity and

logical progression.

Keyword Relevance:

Evaluating the relevance of keywords extracted from the summary involves assessing their alignment with the main topics discussed in the video. This assessment considers both the frequency of occurrence of important terms and their contextual relevance within the summary. High-frequency keywords that accurately reflect the central themes of the video are deemed more relevant. Additionally, contextual relevance is crucial, as it ensures that keywords are used appropriately and reflect the content accurately. By analyzing these aspects, keyword metrics provide valuable insights into the effectiveness of summarization in capturing and highlighting the essential elements of the video, facilitating a comprehensive understanding of its content.

Length and Compression Ratio:

Comparing the length of the summary to that of the original transcript offers valuable insights into the summarization process's efficacy. A higher compression ratio, achieved by significantly reducing the length of the original transcript while retaining crucial information, indicates a more concise summary. This compression ensures that extraneous details are omitted, leading to a streamlined and focused summary. However, maintaining the essential information within the condensed format is crucial to ensure the summary remains informative and comprehensive. By analyzing the compression ratio, we can assess how effectively the summary captures the core content of the video, balancing brevity with informativeness to provide a succinct yet meaningful overview.

6.3 Testing Graphical User Interfaces:

YTV Summarizer can confidently launch its graphical user interface (GUI), it must undergo rigorous testing to ensure functionality, compatibility, accessibility, performance, and user acceptance. Beginning with functionality testing, each interactive element, including buttons, input fields, and navigation menus, will be scrutinized to verify that they perform their intended functions flawlessly. Following this, compatibility testing will be conducted to ensure that the GUI functions seamlessly across various devices, screen sizes, and web browsers, addressing any compatibility issues that may arise on different platforms

Functionality Testing: Verifying that all interactive elements of the GUI, such as buttons, input

fields, and navigation menus, perform their intended functions correctly.

Compatibility Testing: Ensuring that the GUI functions correctly across different devices, screen sizes, and web browsers. This involves testing on various platforms to identify and address any compatibility issues.

Accessibility Testing: Assessing the accessibility of the GUI to ensure it is usable by individuals with disabilities. This includes testing for compliance with accessibility standards such as WCAG (Web Content Accessibility Guidelines).

Performance Testing: Evaluating the performance of the GUI, such as its loading speed and responsiveness, to ensure a smooth and seamless user experience.

User Acceptance Testing: Inviting representative users to test the GUI in a real-world environment and gathering feedback on their satisfaction, preferences, and any areas for improvement.

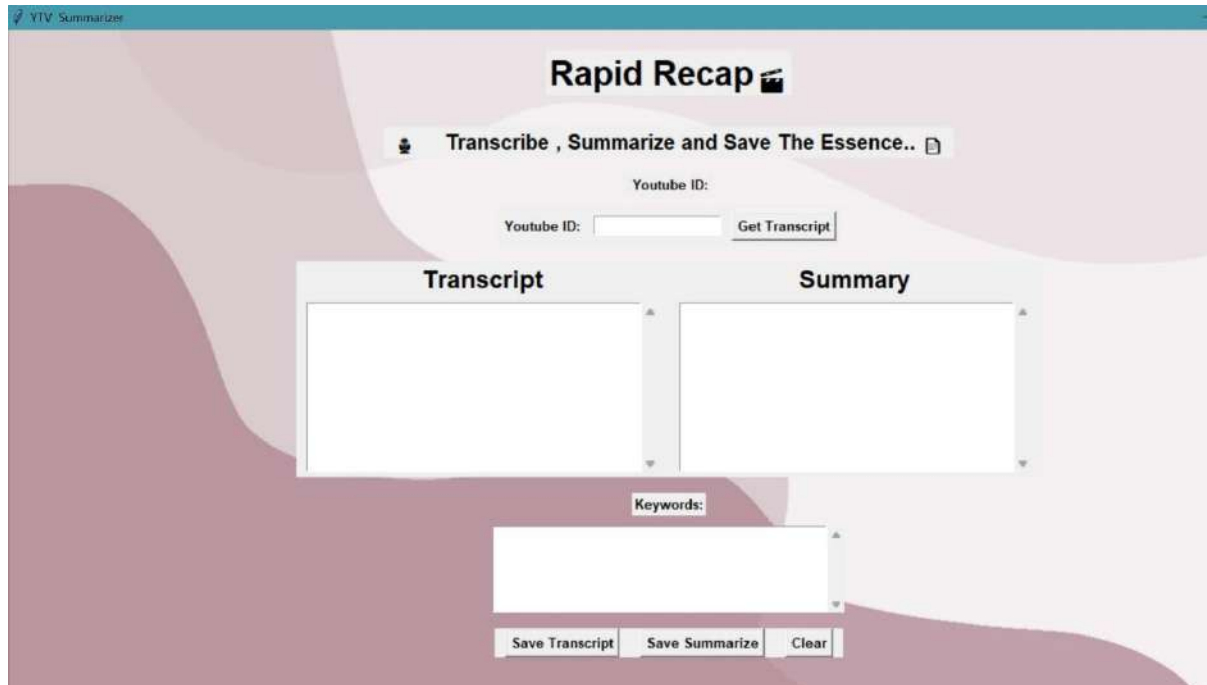


Figure 6.3

CHAPTER-07

CONCLUSION

In conclusion, YTV Summarizer stands as a versatile and user-friendly tool for transcribing and summarizing YouTube videos with efficiency and accuracy. Throughout the user manual, users are provided with clear instructions on installation, usage, customization, and troubleshooting, ensuring a seamless experience from start to finish. By following the outlined steps, users can easily harness the capabilities of YTV Summarizer to obtain concise summaries of video content, enhancing their productivity and understanding. With its intuitive interface, customizable parameters, and comprehensive support resources, YTV Summarizer empowers users to effectively extract key insights from YouTube videos, making it a valuable asset for content creators, researchers, educators, and anyone seeking to streamline their video content analysis workflow.

7.1 Summary:

The user manual for YTV Summarizer provides a detailed guide on installing, using, and customizing the application for efficient transcription and summarization of YouTube videos. It begins with installation instructions, followed by an explanation of input/output formats and usage guidelines. Users are empowered to customize summarization parameters to suit their preferences, with troubleshooting guidance available for any issues. Overall, YTV Summarizer offers a user-friendly solution for extracting key insights from YouTube videos, enhancing productivity across various domains.

7.2 Limitations:

Language Support: YTV Summarizer currently supports only English-language YouTube videos. Expanding language support to include other languages would broaden its usability.

Transcription Accuracy: The accuracy of the transcription heavily depends on the quality of the audio in the YouTube video. YTV Summarizer may struggle with videos featuring poor audio quality or strong accents.

Summarization Quality: While YTV Summarizer provides a useful summary, the quality of the summary may vary depending on the complexity and length of the video content. Improving

the summarization algorithm to handle diverse content more effectively could enhance the overall user experience.

Dependency on External APIs: YTV Summarizer relies on external APIs such as the YouTube Transcription API. Any changes or disruptions to these APIs could impact the functionality of the application.

7.3 Future Work:

Multilingual Support: Expanding YTV Summarizer's language support to include languages other than English would increase its accessibility and usefulness to a broader audience.

Enhanced Summarization Algorithms: Developing more advanced summarization algorithms, such as abstractive summarization techniques, could improve the quality and accuracy of the summaries generated by YTV Summarizer.

Real-time Transcription: Implementing real-time transcription capabilities would allow users to transcribe and summarize live YouTube streams, expanding the application's functionality beyond pre-recorded videos.

User Feedback Integration: Incorporating mechanisms for users to provide feedback on the quality of transcriptions and summaries would enable continuous improvement of YTV Summarizer through user-driven enhancements.

REFERENCES

- [1] IJCRT.ORG. “YOUTUBE TRANSCRIPT SUMMARIZER.” Ijcrt.org, Gousiya Begum , N. Musrat Sultana , Dharma Ashritha, 6 June 2022, <https://ijcrt.org/papers/IJCRT22A6393.pdf>. Accessed 30 March 2023.
- [2] Analytic Vidya. “Creating a Youtube Summariser - Mini NLP Project.” Analytics Vidhya, Basil Saji, 13 January 2022, <https://www.analyticsvidhya.com/blog/2022/01/youtube-summariser-mini-nlp-project/>. Accessed 30 March 2023.
- [3] Rice, Damien, and Matt Galbraith. Video Transcript Summarizer, Atluri Naga Sai Sri Vybhavi, Laggiseti Valli Saroja, Jahnavi Duvvuru, JayanaBayana, 16 November 2008, <https://ieeexplore.ieee.org/document/9751991>. Accessed 30 March 2023.
- [4] “YouTube Transcript Summarizer using Natural Language Processing.” International Journal of Advanced Research in Science, Communication and Technology, <https://ijarsct.co.in/Paper3034.pdf>. Accessed 31 March 2023.
- [5] Gousiya Begum, N. Musrat Sultana, Dharma Ashritha, “YouTube Transcript Summarizer.”, 2022.
- [6] I .Keskes, M. M. Boudabous, M. H. Maaloul, and L. H. Belguith, “Étude comparative entre trois approches de résumé automatique de documents arabes,” in Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, 2012, pp. 225–238.
- [7] H. Froud, A. Lachkar, and S. A. Ouatik, “Arabic text summarization based on latent semantic analysis to enhance arabic documents clustering hanane,” arXiv preprint arXiv, 2013, doi: 10.48550/arXiv.1302.1612
- [8] E. Mahdipour, “Automatic persian text summarizer using simulated annealing and genetic algorithm,” International Journal of Intelligent Information Systems, vol. 3, no. 6, p. 84, 2014, doi: 10.11648/j.ijis.s.2014030601.26.
- [9] F. Kiyoumars, “Evaluation of automatic text summarizations based on human summaries,” Procedia - Social and Behavioral Sciences, vol. 192, pp. 83–91, Jun. 2015, doi: 10.1016/j.sbspro.2015.06.013.

[10] S. Sah, S. Kulhare, A. Gray, S. Venugopalan, E. Prud'hommeaux, and R. Ptucha, "Semantic text summarization of long videos," in Proceedings-2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, 2017, pp. 989–997, doi: 10.1109/WACV.2017.115.

[11] N. L. Garlic, "Computational text analysis of Youtube video transcripts - Loretta C. Duckworth scholars studio," sites.temple.edu, 2019. [Online] <https://sites.temple.edu/tudsc/2019/04/03/computational-text-analysis-of-youtube-video-transcripts/> (accessed Jun. 24, 2021).

APPENDIX-A

SOURCE CODE

```
import tkinter as tk

from tkinter import scrolledtext, messagebox

from youtube_transcript_api import YouTubeTranscriptApi as yta

from reportlab.lib.pagesizes import letter

from reportlab.pdfgen import canvas

from sumy.parsers.plaintext import PlaintextParser

from sumy.nlp.tokenizers import Tokenizer

from sumy.summarizers.lsa import LsaSummarizer

from yake import KeywordExtractor


class YoutubeTranscriberApp:

    def __init__(self):

        self.root = tk.Tk()

        self.root.title("YTV Summarizer")

        self.root.geometry("900x600")

        # Load the background image

        self.bg_image = tk.PhotoImage(file="bg25.png") # Change "background_image.png" to
your image file

        self.background_label = tk.Label(self.root, image=self.bg_image)

        self.background_label.place(relwidth=1, relheight=1)

        # self.create_database() # Remove this line if not using any database

        self.create_widgets()

    def create_widgets(self):
```

```

title_font = ("Helvetica", 25, "bold")
smaller_title_font = ("Helvetica", 18, "bold")

# Title label

self.title_label = tk.Label(self.root, text="Rapid Recap 🎬 ", font=title_font)
self.title_label.pack(pady=20)

t_font = ("Helvetica", 15, "bold")

self.author_label = tk.Label(self.root, text=" 🗣️ Transcribe , Summarize and Save The
Essence.. 📄 ", font=t_font)

self.author_label.pack(pady=10)

m_font = ("Helvetica", 10, "bold")

self.youtube_id_label = tk.Label(self.root, text=" Youtube ID:", font=m_font)
self.youtube_id_label.pack(pady=5)

# Frame for YouTube ID and Keywords

youtube_keywords_frame = tk.Frame(self.root)
youtube_keywords_frame.pack(pady=5)

# YouTube ID Label and Entry

m_font = ("Helvetica", 10, "bold")

self.youtube_id_label = tk.Label(youtube_keywords_frame, text="Youtube ID:",
font=m_font)

self.youtube_id_label.pack(side=tk.LEFT, padx=5, pady=5)

self.youtube_id_entry = tk.Entry(youtube_keywords_frame)
self.youtube_id_entry.pack(side=tk.LEFT, padx=5, pady=5)

# Get Transcript Button

```

```

g_font = ("Helvetica", 10, "bold")

self.transcript_button = tk.Button(youtube_keywords_frame, text="Get Transcript",
command=self.get_transcription, font=g_font)

self.transcript_button.pack(side=tk.LEFT, padx=5, pady=5)


# Frames for Transcript and Summary
text_frames = tk.Frame(self.root)
text_frames.pack(pady=10)


# Transcript Frame with Heading
transcript_frame = tk.Frame(text_frames)
transcript_frame.pack(side=tk.LEFT, padx=10)


transcript_title_label = tk.Label(transcript_frame, text="Transcript",
font=smaller_title_font)
transcript_title_label.pack()


# Transcript Text Box
self.transcript_text = scrolledtext.ScrolledText(transcript_frame, wrap=tk.WORD,
width=40, height=10)
self.transcript_text.pack(pady=5)


# Summary Frame with Heading
summary_frame = tk.Frame(text_frames)
summary_frame.pack(side=tk.RIGHT, padx=10)


summary_title_label = tk.Label(summary_frame, text="Summary",
font=smaller_title_font)
summary_title_label.pack()


# Summary Text Box

```

```

        self.summary_text = scrolledtext.ScrolledText(summary_frame, wrap=tk.WORD,
width=40, height=10)

        self.summary_text.pack(pady=5)

# Keywords Label and Text Box
        self.keywords_label = tk.Label(self.root, text="Keywords:", font=m_font)
        self.keywords_label.pack(pady=5)

        self.keywords_text = scrolledtext.ScrolledText(self.root, wrap=tk.WORD, width=40,
height=5)
        self.keywords_text.pack(pady=5)

# Buttons Frame
        buttons_frame = tk.Frame(self.root)
        buttons_frame.pack(pady=10)

# Save Transcript Button
        self.save_pdf_button = tk.Button(buttons_frame, text="Save Transcript",
command=self.save_as_pdf, font=g_font)
        self.save_pdf_button.pack(side=tk.LEFT, padx=10)

# Save Summaries Button
        self.save_summaries_button = tk.Button(buttons_frame, text="Save Summaries",
command=self.save_summaries_line_by_line, font=g_font)
        self.save_summaries_button.pack(side=tk.LEFT, padx=10)

# Clear Button
        self.clear_button = tk.Button(buttons_frame, text="Clear", command=self.clear_all,
font=g_font)
        self.clear_button.pack(side=tk.LEFT, padx=10)

```

```

def get_transcription(self):
    youtube_id = self.youtube_id_entry.get()
    if not youtube_id:
        messagebox.showinfo("Error", " ❌ Please enter a valid Youtube ID.")
        return

    try:
        data = yta.get_transcript(youtube_id)
        transcript = ""
        for value in data:
            for key, val in value.items():
                if key == 'text':
                    transcript += val + '\n'

        self.transcript_text.delete(1.0, tk.END)
        self.transcript_text.insert(tk.END, transcript)

        # Generate video summary and extract keywords
        self.generate_summary(transcript)

        # Save data to the database
        youtube_link = f"https://www.youtube.com/watch?v={youtube_id}"
        # self.insert_into_database(youtube_id, youtube_link, transcript,
        self.summary_text.get(1.0, tk.END)) # Uncomment this if you have a database

    except Exception as e:
        messagebox.showinfo("Error", f" ❌ An error occurred: {str(e)}")

def generate_summary(self, transcript):

```

```

parser = PlaintextParser.from_string(transcript, Tokenizer("english"))
summarizer = LsaSummarizer()
summary_sentences = summarizer(parser.document, 3)
summary = ' '.join(str(sentence) for sentence in summary_sentences)

# Extract keywords using YAKE
extractor = KeywordExtractor()
keywords = extractor.extract_keywords(text=transcript)
keywords_text = '\n'.join([keyword for keyword, _ in keywords])

# Append keywords to the end of the summary
summary_with_keywords = summary
self.summary_text.delete(1.0, tk.END)
self.summary_text.insert(tk.END, summary_with_keywords)

self.keywords_text.delete(1.0, tk.END)
self.keywords_text.insert(tk.END, keywords_text)

def save_summaries_line_by_line(self):
    summary_text = self.summary_text.get(1.0, tk.END)

    if not summary_text.strip():
        messagebox.showinfo("Error", " ❌ Summary is empty. Please generate a summary first.")
        return

    self.save_summaries_as_pdf()

def save_as_pdf(self):

```



```

transcript_text = self.transcript_text.get(1.0, tk.END)

if not transcript_text.strip():
    messagebox.showinfo("Error", "✖ Transcript is empty. Please get the transcript
first.")
    return

pdf_filename = "transcript.pdf"
c = canvas.Canvas(pdf_filename, pagesize=letter)
width, height = letter
lines = transcript_text.split('\n')
y_position = height - 100

for line in lines:
    c.drawString(100, y_position, line)
    y_position -= 12

c.save()

messagebox.showinfo("Success", f"✅ Transcript saved as {pdf_filename}")

# Save data to the database
youtube_id = self.youtube_id_entry.get()
youtube_link = f"https://www.youtube.com/watch?v={youtube_id}"

# self.insert_into_database(youtube_id, youtube_link, transcript_text,
self.summary_text.get(1.0, tk.END)) # Uncomment this if you have a database

def save_summaries_as_pdf(self):
    summary_text = self.summary_text.get(1.0, tk.END)
    if not summary_text.strip():

```

```
        messagebox.showinfo("Error", "✖ Summary is empty. Please generate a summary first.")
```

```
    return
```

```
pdf_filename = "summaries.pdf"
```

```
c = canvas.Canvas(pdf_filename, pagesize=letter)
```

```
width, height = letter
```

```
lines = summary_text.split('\n')
```

```
y_position = height - 100
```

```
for line in lines:
```

```
    c.drawString(100, y_position, line.strip()) # Print each line of summary
```

```
    y_position -= 12
```

```
c.save()
```

```
messagebox.showinfo("Success", f"✅ Summaries saved as {pdf_filename}")
```

```
def clear_all(self):
```

```
    self.transcript_text.delete(1.0, tk.END)
```

```
    self.summary_text.delete(1.0, tk.END)
```

```
    self.keywords_text.delete(1.0, tk.END)
```

```
def __del__(self):
```

```
    pass
```

```
    # self.connection.close() # Uncomment this if you have a database
```

```
if __name__ == "__main__":
```

```
    app = YoutubeTranscriberApp()
```

```
    app.root.mainloop()
```

To view the contents stored in a database using Python.

```
import sqlite3

conn = sqlite3.connect("transcriber_database.db")
cursor = conn.cursor()

# Execute SQL queries
cursor.execute("SELECT * FROM videos;")
print(cursor.fetchall())

# Close the connection
conn.close()
```

APPENDIX-B

OUTPUT SCREENSHOT

YTV Summarizer

Rapid Recap 🎬

🔊 Transcribe , Summarize and Save The Essence.. 📄

Youtube ID:

Youtube ID:

Transcript

Summary

Keywords:

YTV Summarizer

Rapid Recap 🎬

🔊 Transcribe , Summarize and Save The Essence.. 📄

Youtube ID:

Youtube ID:

Transcript

▶

Number nine, AI for early detection of sepsis. Considered a global health crisis, sepsis claims a life every 2.8 seconds. Early diagnosis is critical, but can be complicated because symptoms are common among other conditions.

Summary

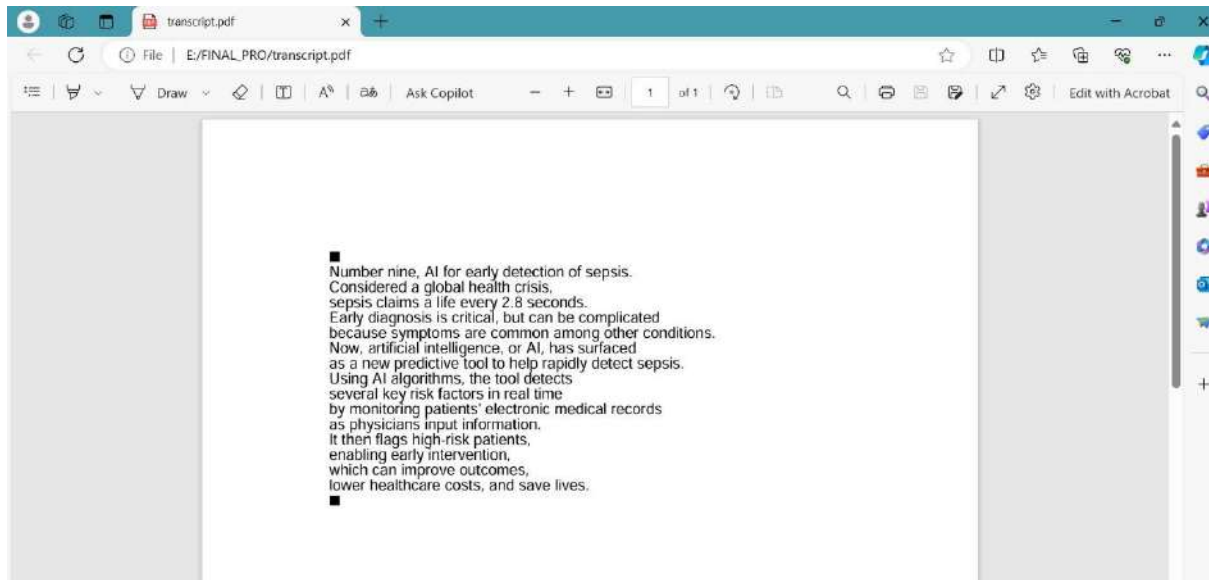
Now, artificial intelligence, or AI, has surfaced as a new predictive tool to help rapidly detect sepsis. Using AI algorithms, the tool detects several key risk factors in real time by monitoring patients' electronic medical records as physicians input information. It then flags high-risk patients, enabling early intervention, which can improve outcomes, lower healthcare costs, and

Keywords:

sepsis
early
tool
Number
Considered a global

After transcribing and summarizing the data, it has been saved as a PDF document using Python.

File Name:Transcript.pdf



File Name : summaries.pdf

