TRANSFORM PLAY BUTTON TO TEXT WITH ADVANCED AI

Mini Project Report

Submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology (B.Tech)

in

COMPUTER SCIENCE AND ENGINEERING

By

B. SREE KAVYA SUDHA	**21AG1A05D9**
M. CHANDRA PRIYA	**21AG1A05G4**
MD. RASHIDA TAHSEEN	**21AG1A05G7**
S. SINDHU	**2A1G1A05I5**

Under the Esteemed Guidance of
Dr. M. V. Vijaya Saradhi
Professor and Dean Of Computer Science and Engineering

Department of Computer Science and Engineering

ACE ENGINEERING COLLEGE

An AUTONOMOUS Institution

NBA Accredited B. Tech Courses, Accorded NAAC 'A' Grade

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telangana)

Ankushapur (V), Ghatkesar (M), Medchal – Malkajgiri Dist - 501 301.

FEBRUARY 2025

ACE
Engineering College
An AUTONOMOUS Institution
Website: www.aceec.ac.in E-mail: info@aceec.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project work entitled **"TRANSFORM PLAY BUTTON TO TEXT WITH ADVANCED AI"** is being submitted by **B.SREE KAVYA SUDHA(21AG1A05D9), M.CHANDRA PRIYA (21AG1A05G4), MD.RASHIDA TAHSEEN (21AG1A05G7), S.SINDHU (21AG1A05I5)** in partial fulfillment for the award of Degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** to the Jawaharlal Nehru Technological University, Hyderabad during the academic year 2024-25 is a record of bonafide work carried out by them under our guidance and supervision.

The results embodied in this report have not been submitted by the student to any other University or Institution for the award of any degree or diploma.

Internal Guide	**HoS**	**Dean-CSE**
Dr.M.V.Vijaya Saradhi	**Mr.V.Chandra Sekhar Reddy**	**Dr.M.V.Vijaya Saradhi**

Project Coordinator
Mrs. Ch. Srivatsa

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our gratitude to all the people behind the screen who have helped us transform an idea into a real time application.

We would like to express our heart-felt gratitude to our parents without whom We would not have been privileged to achieve and fulfill our dreams.

A special thanks to our General Secretary, **Prof. Y. V. Gopala Krishna Murthy,** for having founded such an esteemed institution. Sincere thanks to our COO **Mr. Y.V.Raghu Vamshi**, for support in doing project work. We are also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project. We are very much thankful to our beloved Vice Principal & Dean-Skill Development **Dr. M. Murali** for his continuous encouragement to fulfill our project.

We profoundly thank **Dr. M. V. Vijaya Saradhi**, Professor and Dean of Computer Science and Engineering, who has been an excellent guide and also a great source of inspiration to our work.

We extremely thank **Mr. V Chandra Sekhar Reddy**, HoS & Associate Professor, who helped us in all the way in fulfilling of all aspects in completion of our Mini-Project.

We sincerely thank **Mrs. Ch. Srivatsa**, Assistant Professor, for her continuous support and guidance throughout the project.

We are very thankful to our internal guide **Dr. M. V. Vijaya Saradhi**, Professor and Dean Of Computer Science and Engineering, who has been an excellent and also given continuous support for the Completion of our project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, We would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased our task.

By

B.SREE KAVYA SUDHA (21AG1A05D9)

M.CHANDRA PRIYA (21AG1A05G4)

MD.RASHIDA TAHSEEN (21AG1A05G7)

S.SINDHU (21AG1A05I5)

DECLARATION

We hereby declare that this mini project entitled "**TRANSFORM PLAY BUTTON TO TEXT WITH ADVANCED AI**" submitted to the ACE Engineering College, is a record of an original work done by us under the guidance of **Dr. M. V. VIJAYA SARADHI**, Professor and Dean of Computer Science and Engineering, **ACE Engineering College**, and this project work submitted in the partial fulfillment of the requirements for the mini project; the results embodied in this thesis have not been submitted to any other university or institute for award of any degree or diploma.

B.SREE KAVYA SUDHA (21AG1A05D9)
M.CHANDRA PRIYA (21AG1A05G4)
MD.RASHIDA TAHSEEN (21AG1A05G7)
S.SINDHU (21AG1A05I5)

INDEX

ABSTRACT

Video summarization is a process that condenses lengthy videos into shorter, more concise versions, retaining the most important content. It involves selecting key frames, segments, or extracting textual transcripts to create a coherent summary. These summaries serve as a timesaving way to access essential information from videos, making it easier for users to quickly understand the video's content without watching it in its entirety. Machine learning plays a pivotal role in video summarization. Algorithms, particularly deep learning models like Long Short-Term Memory (LSTM) networks, are employed to automatically identify significant moments within a video. Machine learning models analyze visual and audio cues, speaker sentiment, and transcript text to determine the most relevant segments. These segments are then stitched together to form a comprehensive and coherent summary, making video summarization an efficient and accessible means to extract valuable insights from videos, all driven by intelligent algorithms and neural networks.

In addition to video summarization, another critical aspect of this project is the translation of English text into various Indian regional languages, including Hindi, Bengali, Tamil, Telugu, Marathi, Gujarati, Kannada, and Malayalam. To facilitate this translation process, I utilized Google Translate, which provides a robust framework for converting English text into these languages accurately.

This component of the project aims to make video content more accessible to a broader audience by providing summaries in native languages. By integrating translation capabilities, users can engage with the summarized content in their preferred language, enhancing comprehension and ensuring that linguistic barriers do not hinder access to important information. The choice of languages reflects the linguistic diversity of India, allowing speakers of different regional languages to benefit from the video content seamlessly.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S. No.	Abbrevation	Full Form
1	LSTM	Long Short Term Memory
2	DL	Deep Learning
3	NLP	Natural Language Processing
4	ML	Machine Learning

CHAPTER 1
INTRODUCTION

The proposed project is designed to simplify the process of extracting meaningful information from lengthy YouTube videos by automatically summarizing their content. With the rise in online video consumption, particularly educational and informational content on platforms like YouTube, users often face the challenge of sifting through hours of video to find key insights. This project aims to solve that problem by providing users with concise summaries of YouTube videos, allowing them to quickly grasp the essential points without needing to watch the entire video.

The system utilizes cutting-edge machine learning techniques and natural language processing to generate summaries. The workflow begins with the user providing a YouTube video link. The system then processes the video by extracting the audio, transcribing the spoken content into text using a speech-recognition library, and preprocessing the transcript to ensure clarity and relevance. The heart of the system is an LSTM-based summarization model, which generates a coherent and concise summary of the transcript.

Additionally, the system offers a translation feature, allowing the generated English summary to be converted into the user's native language using Google Translator, ensuring accessibility for non-English speakers. The project provides a user-friendly interface for both inputting the video link and viewing the generated summaries, making it a convenient tool for individuals across various domains who need quick access to key video content.

This innovative approach combines state-of-the-art deep learning techniques with practical applications, offering a valuable solution for users who want to save time and enhance their productivity when consuming video content.

The primary objective of this project is to develop a comprehensive system for video summarization and translation of English text into Indian regional languages. Specifically, the project aims to identify and extract key segments from lengthy videos to create concise summaries that retain essential information. Additionally, it seeks to translate these summaries

into Hindi, Bengali, Tamil, Telugu, Marathi, Gujarati, Kannada, and Malayalam, ensuring accessibility for non-English speakers. By integrating machine learning techniques and translation tools, the project aspires to enhance user engagement, promote inclusivity, and facilitate effective communication across linguistic barriers in a diverse society.

The scope of this project encompasses both video summarization and the translation of English text into various Indian regional languages. It focuses on utilizing machine learning algorithms, particularly deep learning models like LSTM networks, to analyze video content and identify key moments for summarization. The project will cover a diverse range of video types, including educational, informational, and entertainment content, ensuring broad applicability.

In terms of translation, the project will provide translations for summaries into Hindi, Bengali, Tamil, Telugu, Marathi, Gujarati, Kannada, and Malayalam, catering to a wide audience across India. The system aims to maintain the integrity and meaning of the original content during the summarization and translation processes.

Additionally, the project will explore user interfaces that facilitate easy access to summarized and translated content, promoting inclusivity and enhancing the overall user experience. Ultimately, the project aims to empower users by making valuable information accessible regardless of language proficiency.

CHAPTER 2

LITERATURE SURVEY

Recent advancements in video summarization have been greatly influenced by deep learning techniques, significantly improving the accuracy and efficiency of generating video summaries. Early techniques in video summarization, such as keyframe extraction, video skimming, and content-based summarization, provided the foundation for automating the process of selecting important frames or video segments. This study also show recent improvements and the approaches they have followed.

This paper proposes an automatic video summarization algorithm using NLP based algorithms. With an increase in internet videos on the video repository platforms like YouTube, Instagram etc. there is an increase in demand for good summarization algorithms to summarize various videos. This paper aims to produce short and concise video summary that summarizes various YouTube videos.

Text summarization is a technique for extracting concise summaries from a large text without sacrificing any important information. It\'s a good way to extract crucial information from documents. The rapid rise of the internet has resulted in a substantial surge in data all across the world. It has become difficult for humans to manually summarise big documents. Automatic Text Summarization is an NLP technique that lowers the time and efforts required by a human to create a summary.

In their survey, Wan et al(2019) provide a thorough review of video summarization techniques using deep learning. They explore various methods such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid deep learning models for video summarization. The key advantage of these techniques is that they can automatically extract and process features from video data, significantly reducing the need for manual feature extraction. However, the methods come with high computational costs and require large labeled datasets, which can make them challenging to implement at scale.

Recent studies introduce a self-supervised approach for video summarization using transformer networks(2020). This method reduces the dependency on labeled data, making it

more efficient and accessible for large datasets. The self-supervised model allows the network to learn features from the video data itself, promoting adaptability. However, one limitation is that the model struggles with temporal consistency, as it sometimes fails to capture the appropriate flow of events in the video. Additionally, it requires extensive training data to achieve high accuracy, which can be a resource-intensive process.

This approach uses graph-based models to summarize videos(2021), where video segments are represented as nodes in a graph and the relationships between segments are captured as edges. This approach excels at identifying key moments and content relevance, making it efficient for content retrieval. However, it encounters challenges when applied to long videos due to the complexity of managing large graphs. The model's performance can degrade as the size of the video increases, leading to higher computational costs and longer processing times.

This paper(2022) introduces an adversarial learning framework for unsupervised video summarization. In this model, a generator creates video summaries, while a discriminator evaluates their quality, allowing the system to refine its outputs. The key advantage is that this method eliminates the need for labeled data, which is often difficult to obtain. However, adversarial models are challenging to train and can produce variable output quality. The complexity of training these models is a significant drawback, especially when scaling the approach to larger datasets.

Each of these papers represents a different approach to solving the challenges of video summarization, with varying strengths and weaknesses. From reducing the need for labeled data to improving computational efficiency, these techniques reflect the ongoing evolution of video summarization methods, driven by advancements in deep learning, reinforcement learning, and unsupervised learning. While each methodology has its advantages, they also face challenges that hinder their broader adoption and practical implementation, such as high computational costs, the need for large datasets, and issues with training consistency.

Year	Author	Name	Advantages	Disadvantages
2021	Kiran K N	Video Summarization Using NLP	Covers various deep learning methods for summarization.	High computational cost and need for large labeled datasets
2022	Monali Bansode	Extractive Text And Video Summarization Using TF-IDF Algorithm	Reduces dependency on labeled data	Struggles with temporal consistency and requires extensive training
2022	Sarah S. Alrumiah	Educational Videos Subtitles Summarization	Efficient in detecting key moments and content relevance	High complexity when applied to long videos
2021	Shraddha Yadav	Summary And Keyboard Extraction From Youtube Video Transcript	Dynamically adjusts summary length based on content	High training time and requires extensive datasets
2020	Madhu S. Nair	Static video Summarization Using Multi CNN With Random Forest Classifier	No need for labeled data, improves summarization quality	Adversarial models are complex to train and have variable output quality
2017	Snehal Hiray	Video Summarization	Enhanced feature extraction and accuracy over standalone models	Requires large-scale datasets and complex training
2023	Tanuja Sachin Khatavkar	Video Summarization using NLP	Improved scene understanding, does not require labeled datasets	Overfitting on small datasets and needs domain adaptation
2021	YoungJin Suh	Deep Learning-Based Video Summarization	Fast event detection, adaptable to sports video analysis	High resource consumption and real-time optimization issues

Table 2.1. Literature Survey

CHAPTER 3

SOFTWARE REQUIREMENT ANALYSIS

3.1. PROBLEM STATEMENT

The rapid growth of video content in various domains has led to an overwhelming amount of information available online. Users often struggle to find essential insights within lengthy videos, resulting in time inefficiency and potential information overload. Moreover, in a diverse country like India, linguistic barriers further complicate access to information. Many users may not be proficient in English, limiting their ability to understand video content that lacks translations. This project addresses two interconnected problems: the need for effective video summarization to condense content and the necessity of translating summaries into multiple Indian regional languages. By focusing on these aspects, the project aims to enhance user engagement and accessibility, enabling a broader audience to benefit from important information. The ultimate goal is to create a seamless experience for users, allowing them to quickly comprehend video content and engage with it in their native language, thus fostering inclusivity and better information dissemination.

3.2. MODULES AND THEIR FUNCTIONALITIES

1. User Module:

The User Module handles user authentication, registration, and session management within the system. It ensures that only registered users can access the summarization and video processing features.

- **User Login:** Authenticates users based on email and password.
- **User Registration:** Allows new users to sign up and stores their data in a MySQL database.
- **Session Management:** Maintains user sessions after login using Flask sessions.
- **Navigation & UI Rendering:** Handles user navigation to different pages like homepage, about page, and user dashboard.

Main Functions:

- **login():** Authenticates users and starts a session.
- **registration():** Registers a new user and saves their details in the database.

- **User home():** Loads the dashboard for logged-in users.
- **index() & about():** Serve static pages for the homepage and about page.

2. Summarization and Translation Module:

This module is responsible for extracting transcripts from YouTube videos, summarizing the text, and translating the summary into different languages. It provides both extractive and abstractive summarization methods.

- **Transcript Extraction:** Retrieves the transcript of a YouTube video using YouTubeTranscriptApi.
- **Text Summarization:**
  Extractive (using gensim.summarization)
- **Summary Translation:** Translates the summarized text into a chosen language using Google Translate.

  Main Functions:
- **get_transcript(video_id):** Fetches the transcript of a YouTube video.
- **get_summary(text, method, percentage):** Summarizes the text using extractive techniques.
- **translate_summary(text, language):** Translates the summary into the specified language.
- **summary():** Flask route to handle summarization and translation requests from the web interface.

3. Video Processing Module:

The Video Processing Module is responsible for handling YouTube video downloads, extracting key video details, and preparing them for further processing. It plays a crucial role in retrieving video content for summarization and translation.

- **Video Downloading:** Uses pytube to download YouTube videos.
- **Extracting Video ID:** Identifies and extracts the unique YouTube video ID from the URL**.**
- **Handling Video Processing Requests:** Manages video-related user requests through the Flask web interface.

  Main Functions:
- **extract_video_id(url):** Extracts the YouTube video ID from a given URL.

CHAPTER 4

SOFTWARE DESIGN

4.1. ARCHITECTURE DIAGRAM

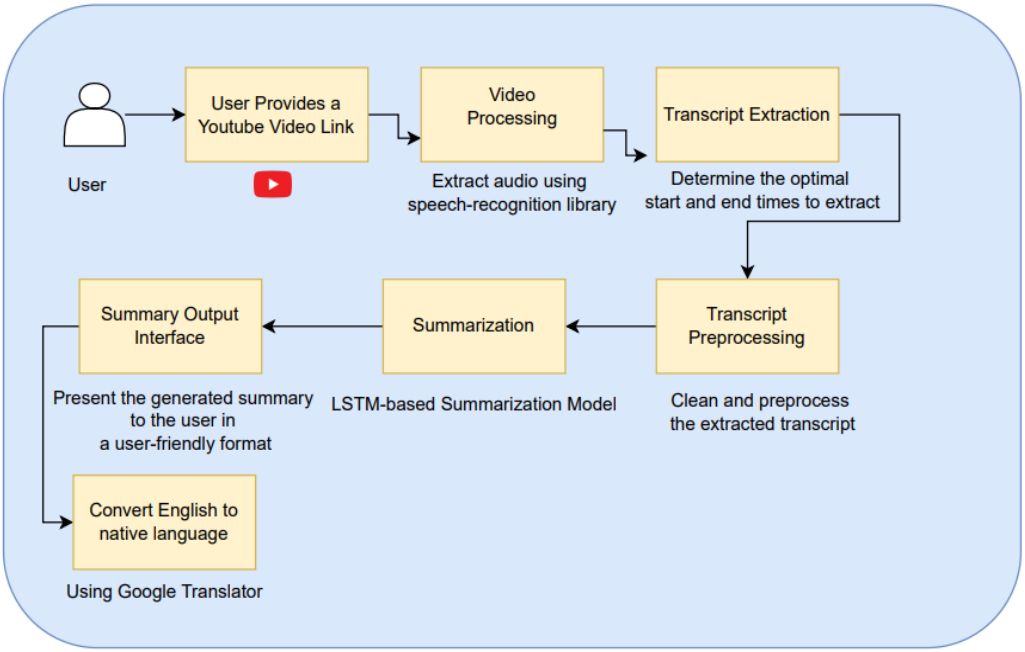

Fig 4. 1. ARCHITECTURE DIAGRAM

1. User Input:

The system begins with the user providing a YouTube video link. This is the starting point where the user interacts with the interface to input the video they wish to summarize. The input format is simply the URL of the video, which is processed in subsequent steps. This user-friendly input mechanism ensures ease of use for individuals of all technical skill levels.

2. Video Processing:

Once the video link is provided, the system processes the video by extracting the audio track from the YouTube video. The audio is isolated using a speech-recognition library, which converts the spoken content in the video into a format suitable for further analysis. This library is capable of handling various accents, background noise, and varying audio quality that may be present in YouTube videos. The video processing module is responsible for ensuring that the audio is accurately captured and prepared for transcription.

3. Transcript Extraction:

After audio extraction, the next step is to transcribe the spoken words into text. Using advanced speech recognition techniques, the system identifies the optimal start and end points of the transcript based on the spoken content of the video. This step is critical as it converts the unstructured audio data into structured text data. The extracted transcript forms the foundation for the summarization process. The accuracy of the transcription significantly impacts the final summary, which is why robust and precise algorithms are used to ensure high-quality text extraction.

4. Transcript Preprocessing:

Following transcript extraction, the text undergoes preprocessing to ensure it is clean and ready for summarization. Preprocessing techniques include removing irrelevant content such as filler words, punctuation marks, and unnecessary stop words. Additionally, this stage may involve correcting grammatical errors and normalizing text for consistency. The cleaned transcript is then structured to be compatible with the machine learning model used for summarization. The preprocessing phase is crucial in improving the performance of the summarization model by providing high-quality, concise input text.

5. Summarization Using LSTM-Based Model:

The core of the system lies in its summarization module, which leverages a Long Short-Term Memory (LSTM) based model. LSTM is a type of recurrent neural network (RNN) that excels in processing sequential data, making it highly suitable for text summarization tasks. The LSTM model analyzes the preprocessed transcript to generate a concise summary of the content, retaining only the most essential points. This model is trained on a variety of datasets to ensure its ability to generalize across different types of video content, from tutorials to interviews and lectures. The use of LSTM ensures that the model captures the context and flow of the video content, providing summaries that are both coherent and informative.

6. Summary Output Interface:

Once the summary is generated, it is displayed in the summary output interface. This interface is designed to be intuitive and user-friendly, allowing users to easily view the condensed version of the video content. The system ensures that the summary retains the key

points of the original content, providing users with a quick overview without needing to watch the entire video. The summary is presented in a format that emphasizes clarity and ease of reading.

7. Translation to Native Language:

For users who prefer to view the summary in their native language, the system integrates Google Translator to convert the English summary into a different language. This feature increases accessibility and ensures that non-English speakers can benefit from the summarization service. The translated summary is presented alongside the original summary, allowing users to compare or directly view the content in their preferred language.

4.2. DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a graphical tool used to visualize the flow of information in a system or process. It uses a standardized set of symbols to represent:

- **Processes:** Transformations that convert data from one form to another.
- **Data Flows:** Movement of data between processes, external entities, and data stores.
- **External Entities:** Sources and destinations of data outside the system.
- **Data Stores:** Places where data is kept for future use.

DFDs are helpful for:

- **Understanding how a system works:** By following the data flow, you can see how data is processed and transformed.
- **Communicating system requirements:** DFDs are a clear and concise way to show what data is needed and how it will be used.
- **Designing new systems:** DFDs can be used to plan out the data flow of a new system before it is built.

4. 2. 1. DFD LEVEL – 0

Level 0 (Context Diagram): A high-level overview of the entire system, showing its main processes and external entities. The below Fig No. 4. 2. 1 shows a high-level overview of the Video Summarization System.

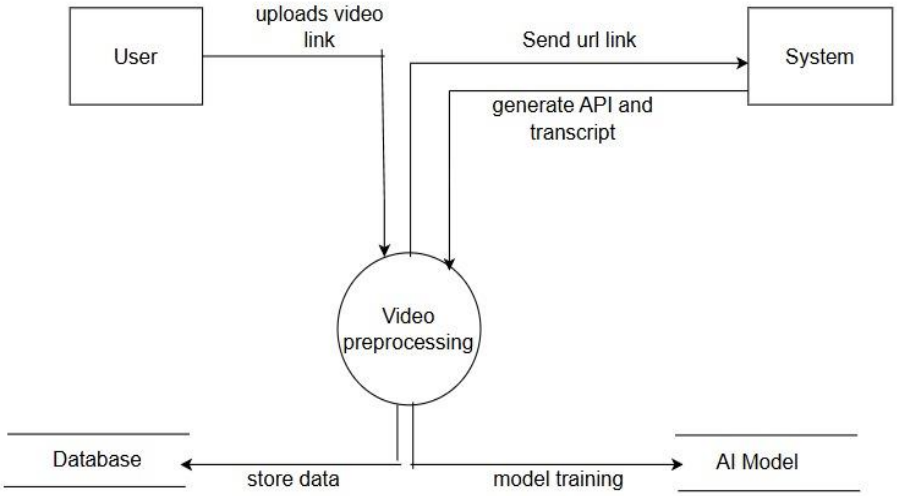

Fig 4. 2. 1. DFD LEVEL – 0

- **User:** Uploads a video link to initiate the summarization process. Interacts with the system to receive processed results.
- **System:** Processes the video URL and generates an API and transcript. Acts as a bridge between the user and video preprocessing.
- **Video Preprocessing:** Extracts key features like frames, audio, and transcripts. Sends processed data to the database and AI model.
- **Database:** Stores processed video data and transcripts. Ensures data availability for future retrieval and analysis.
- **AI Model:** Trains on preprocessed data to generate summaries. Uses machine learning techniques for accurate summarization.

4. 2. 2. DFD LEVEL – 1

Level 1: A more detailed view of a single process, showing its sub-processes and data flows. The below Fig No. 4. 2. 2 shows a high-level overview of the Video Summarization System.

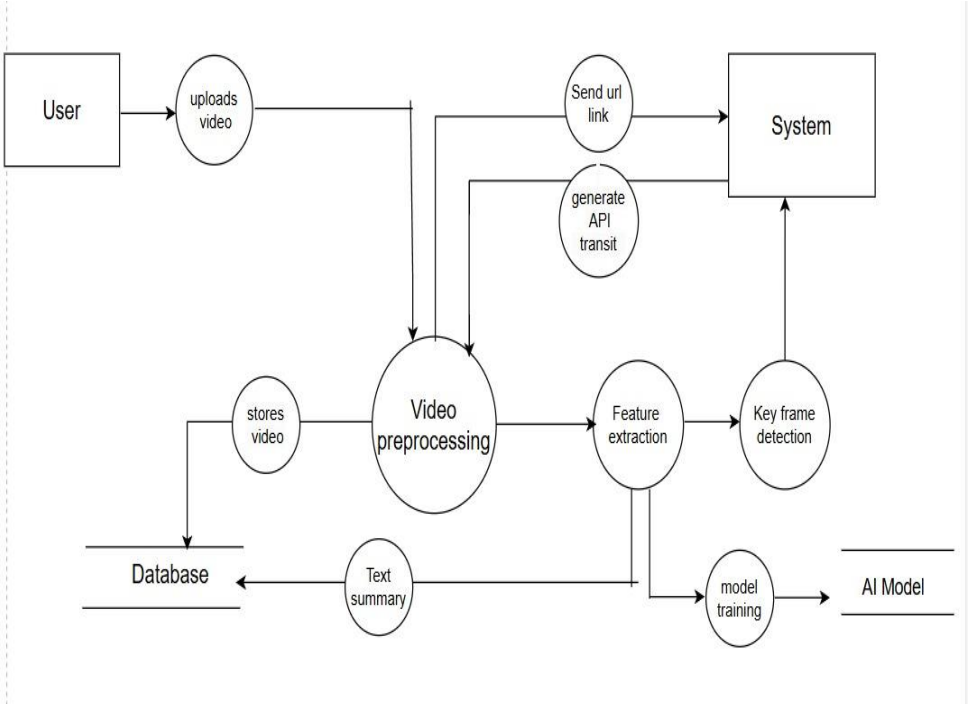

Fig 4. 2. 2. DFD LEVEL – 1

Key Components:

1. **User:** The user uploads a video to the system for processing. This video is then sent for further analysis and storage.
2. **Video Preprocessing:** Handles initial processing of the uploaded video. Prepares the video for feature extraction and AI model training.
3. **System:** Receives the video via a generated URL link. Facilitates communication through API transit for processing.
4. **Feature Extraction:** Extracts essential features from the video content. These features are used for further analysis and model training.
5. **Key Frame Detection:** Identifies and selects important frames from the video. These frames represent crucial moments and are used for AI training.
6. **Model Training:** Uses extracted key frames to train an AI model. This helps improve video analysis, recognition, or summarization tasks.
7. **Text Summary:** Generates a summarized text based on the video content. This can be used for quick reference, reports, or further processing.

4. 3. UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. UML diagrams visually represent a system's structure and behavior. They are divided into **structural** (e.g., Class, Object, Component, Deployment) and **behavioral** (e.g., Use Case, Sequence, Activity, State) diagrams. Structural diagrams define the system's architecture, while behavioral diagrams show interactions and workflows. Use Case Diagrams depict user interactions, and Sequence Diagrams illustrate the flow of processes. These diagrams help in designing, analyzing, and documenting software efficiently.

4. 3. 1. USECASE DIAGRAM

Use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. The below Fig No. 4. 3. 1 can portray the different types of users of a system and the various ways that they interact with the system.

- A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.

- Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

- The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

- A use case diagram visually represents the interactions between users (actors) and a system. It highlights the system's functionality by depicting the use cases that actors can perform. The actors can be people, external systems, or hardware that interact with the system. Each use case represents a specific function or behavior that the system performs in response to an actor's action. Use case diagrams are commonly used in software engineering to gather requirements and illustrate system behaviour.

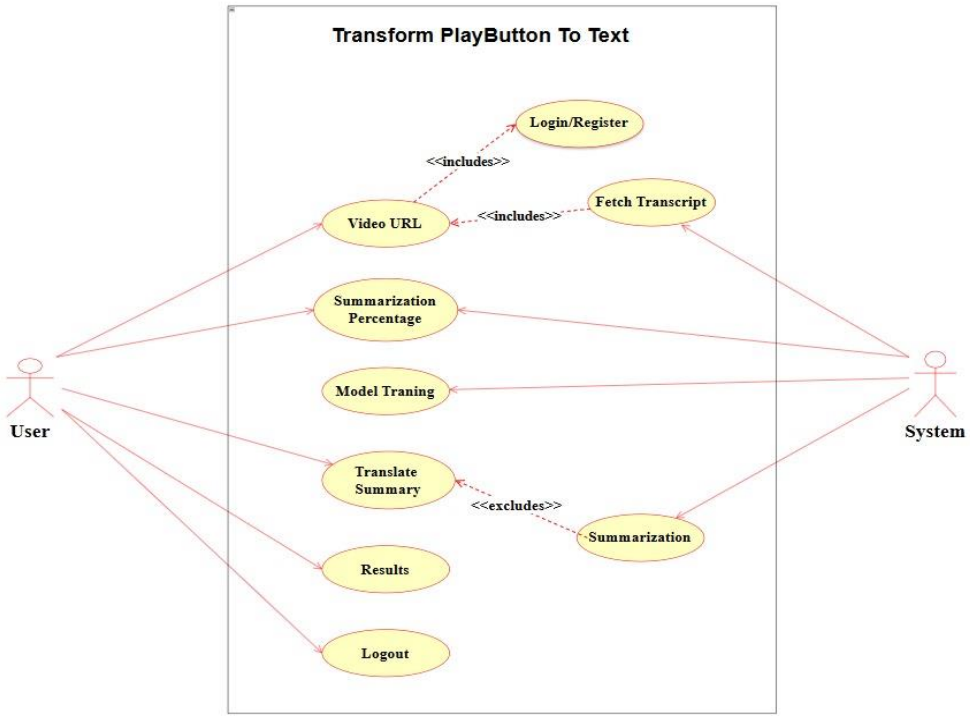

Fig 4. 3. 1. USE CASE DIAGRAM

ACTORS:

1. **User:**
    - Uploads videos and interacts with the summarization system.
    - Adjusts settings like summarization percentage and translation.
    - Views results and logs in/out of the system.
2. **System:**
    - Processes videos for summarization.
    - Fetches transcripts and manages user requests.

USE CASES:

1. **Login/Register:** User authentication before accessing the system.
2. **Video URL:** User provides a video link for processing.
3. **Fetch Transcript:** The system extracts text from the video.
4. **Summarization Percentage:** User selects how much of the video to summarize.
5. **Model Training:** The system improves summarization through training.
6. **Translate Summary:** Optional translation of the summary.
7. **Summarization:** The system generates a video summary.
8. **Results:** User views the summarized output.

4. 3. 2. CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematics of the application and for detailed modeling translating the models into programming code. In the diagram, classes are represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

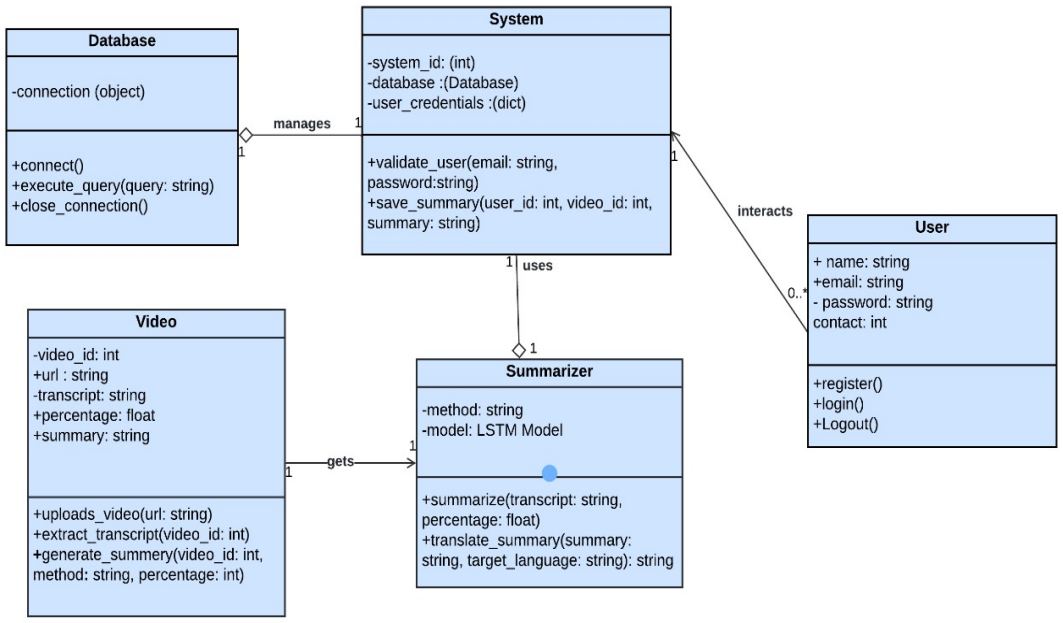

Fig 4. 3. 2. CLASS DIAGRAM

The above Fig No. 4. 3. 2 shows the main components of the system and how they interact with each other. The main components are:

- **Database**: Manages data storage, connections, and query execution. It provides methods for connecting, executing queries, and closing connections.
- **System**: Controls the overall process, validates users, and saves summaries. It interacts with the database and summarization modules.
- **User**: Represents a system user with attributes like name, email, and password. Users can register, log in, and log out.

- **Video**: Stores video-related details like URL, transcript, summary, and percentage. It provides functions to upload, extract transcripts, and generate summaries.
- **Summarizer**: Processes videos using AI/ML models to generate summaries. It also provides translation functionality for summaries.

4. 3. 3 SEQUENCE DIAGRAM

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. A sequence diagram shows the order of interactions between objects in a system over time. It illustrates how messages are exchanged between objects to complete a specific function or process.

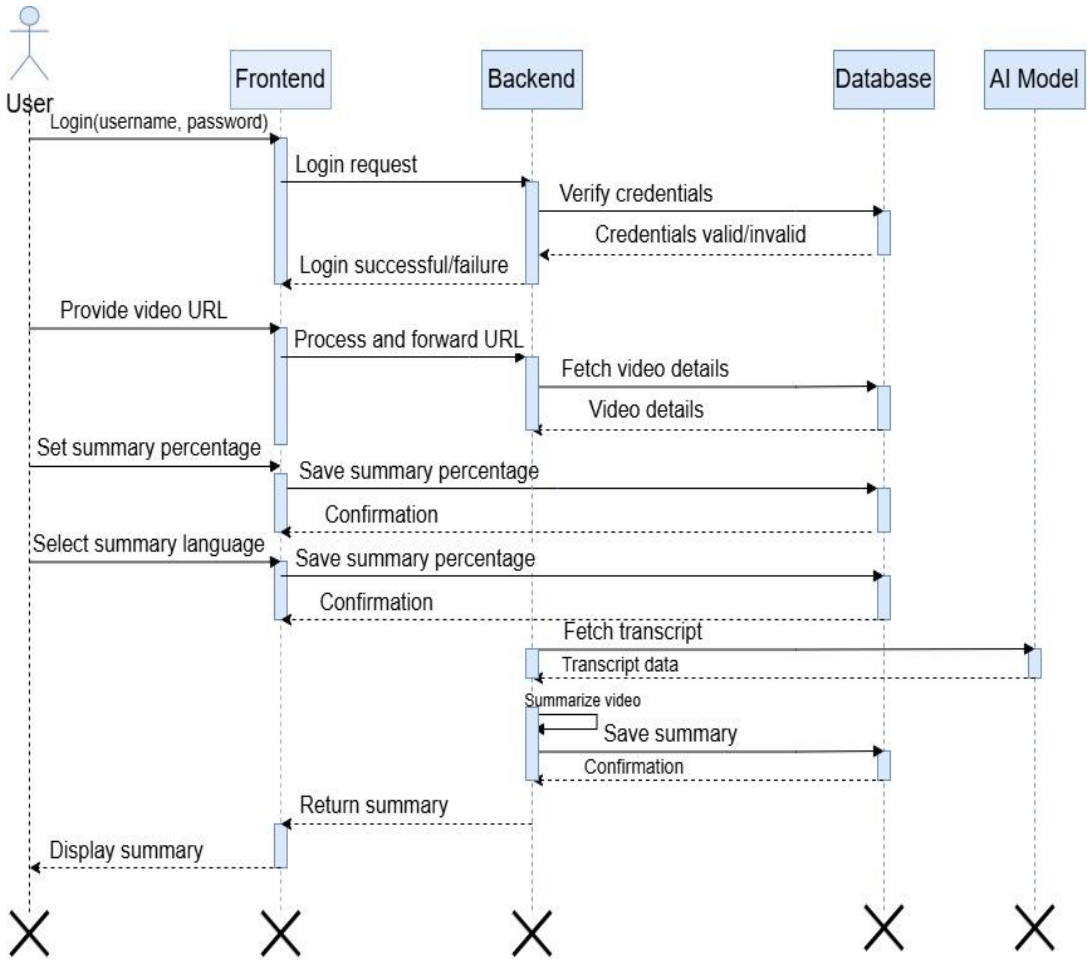

Fig 4. 3. 3. SEQUENCE DIAGRAM

Objects

- **User:** Represents the end user who interacts with the system.
- **Frontend:** The UI/Client interface that communicates with the backend.
- **Backend:** The server-side logic handling authentication, data processing, and AI model interactions.
- **Database:** Stores data such as user credentials, video details, transcripts, and summaries.
- **AI Model:** Performs video transcription and summarization based on input parameters.

Messages

1. **Login:** The user enters credentials.
2. **Login request:** The frontend sends the login request to the backend.
3. **Verify credentials:** Backend checks credentials with the database.
4. **Credentials valid/invalid:** Database responds with authentication status.
5. **Login success/failure:** Backend communicates login results to the frontend.
6. **Provide video URL:** User submits a video link.
7. **Process and forward URL:** Frontend sends the video URL to the backend.
8. **Fetch video details:** Backend retrieves video metadata from the database.
9. **Video details:** Database responds with video information.
10. **Set summary percentage:** User specifies how much of the video to summarize.
11. **Save summary percentage:** Backend stores the summary percentage.
12. **Select summary language:** User chooses the summary language.
13. **Save summary language:** Backend stores the language preference.
14. **Fetch transcript:** Backend requests the video transcript from the database.
15. **Transcript data:** Database sends the transcript.
16. **Summarize video:** Backend calls the AI model with video details, percentage, and language.
17. **Save summary:** Backend stores the generated summary in the database.
18. **Confirmation:** Database acknowledges the saved summary.
19. **Return summary:** Backend retrieves and sends the summary to the frontend.
20. **Display summary:** The frontend presents the summary to the user.

4. 3. 4. COMPONENT DIAGRAM

The below Fig No. 4. 3. 4 depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

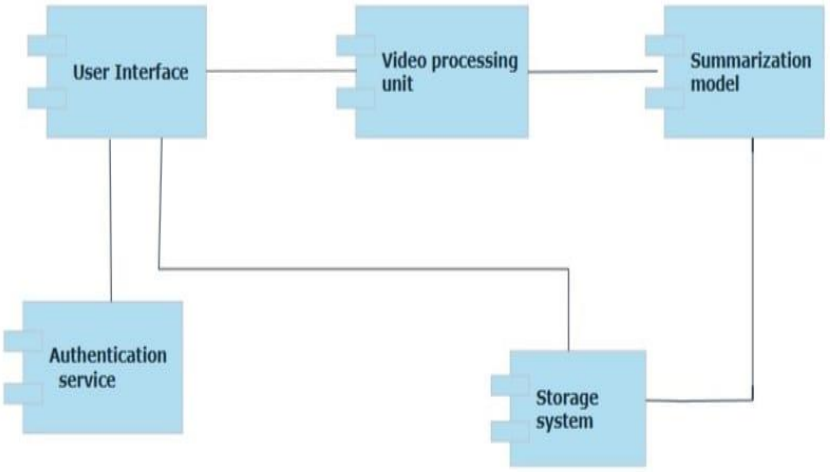

Fig 4. 3. 4. COMPONENT DIAGRAM

User Interface: Enables users to upload videos, configure settings, and view summaries. Connects with authentication and video processing units.

Authentication Service: Manages user authentication and access control for system security. Ensures only authorized users can interact with the system.

Video Processing Unit: Extracts and processes video frames, audio, and metadata. Prepares video content for the summarization model.

Summarization Model: Uses AI/ML algorithms to generate concise summaries from processed video data. Receives input from the video processing unit.

Storage System: Stores original videos, metadata, and generated summaries. Allows retrieval of stored data for future user access.

4. 3. 5. ACTIVITY DIAGRAM

The activity diagram is another important diagram in UML to describe the dynamic aspects of the system. It is a flow chart to represent the flow from one activity to another activity. An **Activity Diagram** is a behavioral UML diagram that visually represents workflows of activities and actions in a system. It consists of nodes (representing actions, decisions, or states) and edges (representing transitions between activities). Activity diagrams help in modeling sequential and parallel processes, decision-making, and control flow. They use symbols like ovals (activities), diamonds (decisions), and arrows (flow of control). These diagrams are widely used in software development to analyze business processes and system functionalities.

The activity can be described as an operation of the system. It shows the process of a user uploading and processing documents in a system as shown in Fig No. 4. 3. 5.

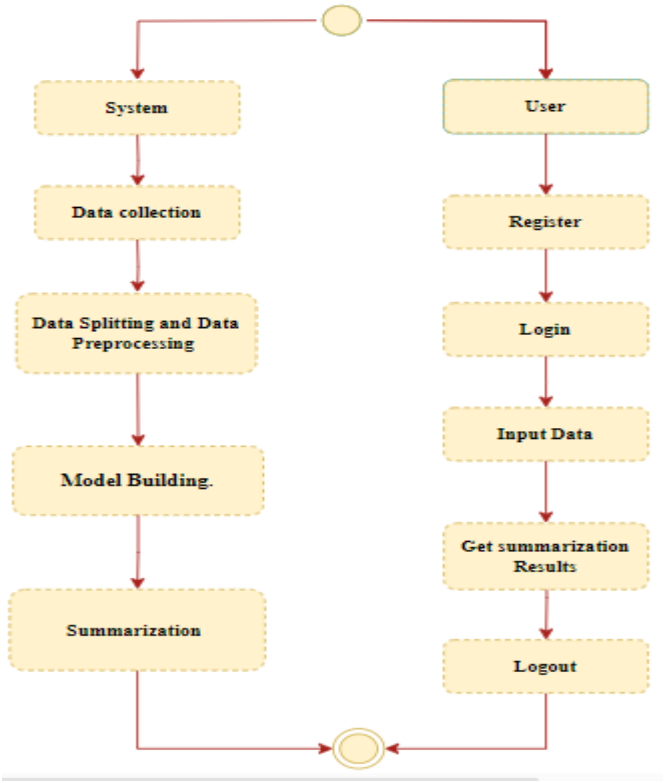

Fig 4. 3. 5. ACTIVITY DIAGRAM

Register: The user creates an account to access the summarization system. Credentials are stored for authentication.

Login: The user provides login details to access the system. The system verifies credentials before granting access.

Input Data: The user submits content for summarization. This data is processed by the system.

Get Summarization Results: The system generates and provides a concise summary based on the input data. The user views or downloads the results.

Logout: The user exits the system, and the session is closed to ensure security.

Data Collection: The system gathers relevant data for training the summarization model. This data can include text, videos, or documents.

Data Splitting and Preprocessing: The collected data is cleaned, tokenized, and divided into training and testing sets for model development.

Model Building: A machine learning or AI model is trained on the processed data to generate accurate summaries.

Summarization: The trained model processes user input and generates a concise summary while maintaining key information.

4. 3. 6. STATE CHART DIAGRAM

A state diagram is a uml diagram which is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State machine diagrams are also known as State Diagrams and State-Chart Diagrams. These both terms can be used interchangeably.

A State Machine Diagram is used to represent the condition of the system or part of the system at finite instances of time. Its a behavioural diagram and it represents the behavior using finite state transitions. In this article, we will explain what is a state machine diagram, the components, and the use cases of the state machine diagram.

This diagram is a State Diagram (or State Machine Diagram) in UML. It shows the different states a Node can be in during a routing process and the transitions between those states.

A State Chart Diagram represents the different states of an object and transitions between them based on events. It includes elements like states, transitions, events, initial and final states, and can model complex behaviors in systems.

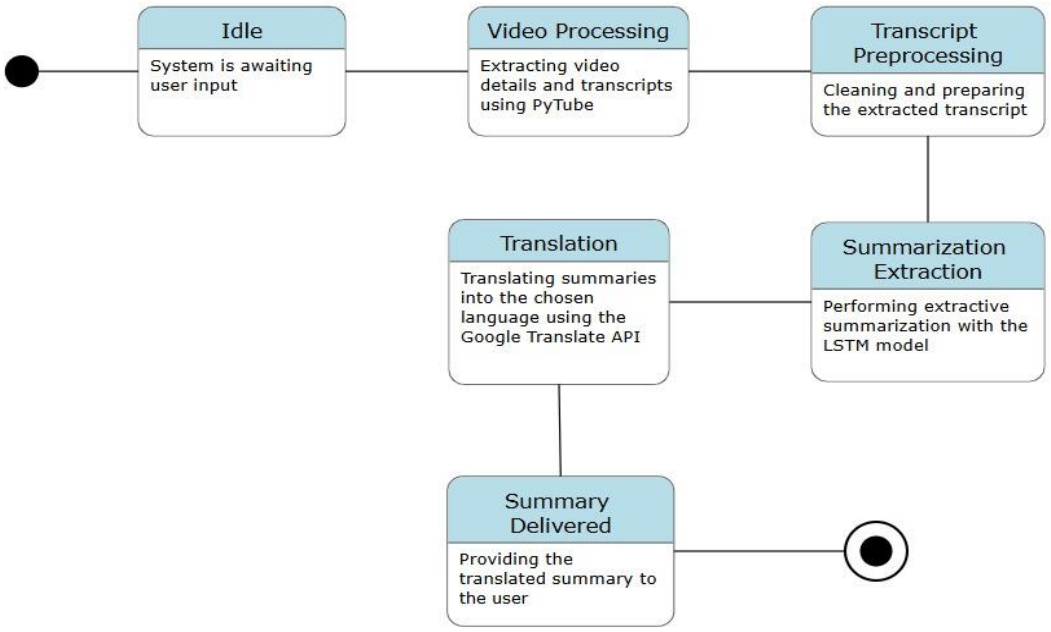

Fig 4. 3. 6. STATE CHART DIAGRAM

States:

The diagram represents a flowchart of a system for summarizing and translating video transcripts. Below is the information for each state in the diagram:

1. **Idle:** The system is in a waiting state, expecting user input to initiate the process.
2. **Video Processing:** The system extracts video details and transcripts using the PyTube library, which is commonly used for working with YouTube videos.
3. **Transcript Preprocessing:** The extracted transcript is cleaned and prepared for further processing, which may involve removing unwanted characters, normalizing text, and structuring it for summarization.
4. **Summarization Extraction:** The preprocessed transcript undergoes extractive summarization using an LSTM (Long Short-Term Memory) model to generate a concise summary.
5. **Translation:** The generated summary is translated into the chosen language using the Google Translate API.
6. **Summary Delivered:** The translated summary is presented to the user as the final output.

Transitions:

The diagram represents a workflow for video summarization and translation. Below is an analysis of the transitions between different states:

1. Idle → Video Processing:

- The system moves from an idle state to processing the video once the user provides input.

2. Video Processing → Transcript Preprocessing:

- After extracting video details and transcripts using PyTube, the system transitions to preprocessing the transcript.

3. Transcript Preprocessing → Summarization Extraction:

- Once the transcript is cleaned and prepared, the system moves to extract a summary using an LSTM model.

4. Summarization Extraction → Translation:

- The generated summary is translated into the desired language using the Google Translate API.

5. Translation → Summary Delivered:

- After translation, the final summary is provided to the user.

6. Summary Delivered → End (Final State):

- The process completes once the summary is delivered.

4. 3. 7. DEPLOYMENT DIAGRAM

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes which is shown in Fig No. 4. 3. 7. A Deployment Diagram is a UML diagram that depicts the physical arrangement of software components on hardware nodes. It illustrates how different servers, cloud instances, client machines, and IoT devices interact to run the system. The diagram includes nodes, artifacts, communication paths, and dependencies, helping in understanding resource allocation and network configurations.

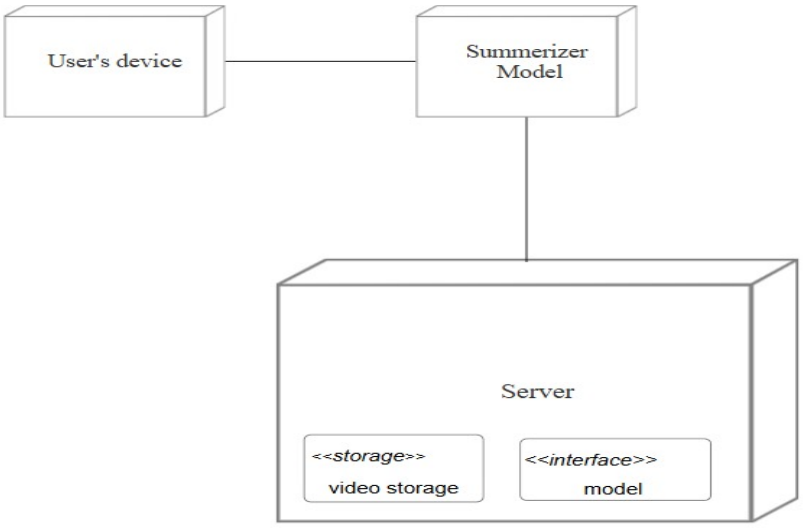

Fig 4. 3. 7. DEPLOYMENT DIAGRAM

User's Device: The user's device interacts with the system to upload videos and receive summarized results. It connects to the summarizer model for processing.

Summarizer Model: An AI-based model that processes video data to generate summaries. It interacts with the server for video input and result generation.

Server: A central unit managing video storage and model execution. It hosts both the video storage and model interface components.

- **Video Storage (Storage Component):** Stores uploaded videos before processing. Ensures accessibility and retrieval for summarization.
- **Model Interface (Interface Component):** Acts as a communication bridge between the summarizer model and server. Facilitates data exchange for video processing.

4. 3. 8. COLLABORATION DIAGRAM

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.

Fig 4. 3. 8. COLLABORATION DIAGRAM

System Actions:

- **YouTube Link** – The user provides a YouTube video link to the system for processing.
- **Transcript Text** – The system extracts the transcript from the video for further analysis.
- **Model Training** – The system trains the summarization model using extracted transcript data.
- **Summarization** – The system generates a summarized version of the video content.

User Actions:

- **Register:** The user creates an account to access the summarization system.
- **Login:** The user logs into the system using credentials to start the process.
- **Input Data:** The user provides input data, such as a video URL, for summarization.
- **Logout:** The user logs out after completing the summarization process.

CHAPTER 5
SOFTWARE AND HARDWARE REQUIREMENTS

5. 1. SOFTWARE REQUIREMENTS

- **IDE/Workbench** : Visual Studio Code.
- **Programming Languages** : Python.
- **Libraries** : numpy, scikit-learn, pandas, MySQL.
- **Operating System** : Windows 11
- **Technology** : Python 3.6+
- **Server Deployment** : Xampp Server

5. 2. HARWARE REQUIREMENTS

- **Processor** - i5/Intel Processor
- **RAM** - Minimum 8GB is recommended
- **Hard Disk** - 160GB
- **Key Board** - Standard Windows Keyboard
- **Mouse** - Two or Three Button Mouse
- **Monitor** - SVGA

CHAPTER 6
MODULES

These are the modules used in our project:

6. 1. User Module

The **User Module** is responsible for managing user authentication and access control within the system. It allows users to register by providing their details such as name, email, password, and contact information. The module securely stores user credentials in a MySQL database and ensures that duplicate registrations are prevented. During login, user credentials are verified against the database, and upon successful authentication, users are granted access to the system. If login fails, appropriate feedback messages are displayed. The module also manages user sessions, ensuring that authenticated users remain logged in during their session and preventing unauthorized access to restricted pages. Flash messages provide users with status updates regarding their registration or login attempts. Additionally, password validation mechanisms are implemented to enhance security and data integrity. After successful login, users are redirected to the homepage where they can access various features. The system also includes error handling to ensure a smooth user experience. A logout functionality is provided to allow users to securely exit their session. This module plays a crucial role in safeguarding user data and ensuring only authenticated users can access the platform.

6. 2. Summarization and Translation Module

The Summarization and Translation Module is responsible for extracting, summarizing, and translating YouTube video content. First, it retrieves the transcript of a YouTube video using the YouTubeTranscriptApi. Once the transcript is extracted, the module provides two summarization methods: extractive summarization (via Gensim). The user can choose between these methods for summarizing the text. After summarization, the module also supports translation of the summary into a specified language using Google Translate.The Flask-based web interface allows users to input a YouTube URL and select their summarization and translation preferences. Once the summary is created and translated, it is displayed on the webpage for the user to view. This module simplifies lengthy videos by condensing the key points and making them available in multiple languages. It is useful for users who want quick,

multilingual summaries of video content. The module handles both text processing and translation seamlessly.

6. 3. Video Processing Module

The Video Processing Module is designed to handle the retrieval and preparation of YouTube video content for further processing. It begins by allowing users to input a YouTube URL. From this URL, the module extracts the video ID, which uniquely identifies the video. The pytube library is then used to download the video, making it ready for tasks like summarization or translation. The module works through a Flask web interface, enabling users to submit video-related requests and interact with the system. Once the video ID is extracted, it can be processed further, such as fetching the transcript or preparing the content for other operations. The module ensures that video data is correctly handled and ready for subsequent steps, ensuring seamless integration with other parts of the system. In summary, the Video Processing Module serves as a foundational component that prepares and processes YouTube videos before any text extraction or manipulation. It facilitates the smooth transition from video URL to video content ready for summarization and translation.

CHAPTER 7

IMPLEMENTATION

7. 1. User Module Implementation

```
# user_module.py

from flask import Flask, render_template, request, session, flash

import mysql.connector


# Database connection

db = mysql.connector.connect(user="root", password="", port='3306',
database='video')

cur = db.cursor()


app = Flask(__name__)

app.secret_key = "###################################"


@app.route('/')

def index():

    return render_template('index.html')


@app.route('/about')

def about():

    return render_template('about.html')


@app.route('/login', methods=['POST', 'GET'])

def login():

    if request.method == 'POST':

        useremail = request.form['useremail']

        session['useremail'] = useremail
```

```
        userpassword = request.form['userpassword']
        sql = "SELECT * FROM user WHERE Email=%s AND Password=%s"
        cur.execute(sql, (useremail, userpassword))
        data = cur.fetchall()
        db.commit()

        if not data:
            msg = "User credentials are not valid"
            return render_template("login.html", name=msg)
        else:
            return render_template("userhome.html", myname=data[0][1])

    return render_template('login.html')

@app.route('/registration', methods=["POST", "GET"])
def registration():
    if request.method == 'POST':
        username = request.form['username']
        useremail = request.form['useremail']
        userpassword = request.form['userpassword']
        conpassword = request.form['conpassword']
        Age = request.form['Age']
        contact = request.form['contact']

        if userpassword == conpassword:
            sql = "SELECT * FROM user WHERE Email=%s AND Password=%s"
            cur.execute(sql, (useremail, userpassword))
            data = cur.fetchall()
            db.commit()
```

```
        if not data:
            sql = "INSERT INTO user (Name, Email, Password, Age, Mob) VALUES
(%s, %s, %s, %s, %s)"
            val = (username, useremail, userpassword, Age, contact)
            cur.execute(sql, val)
            db.commit()
            flash("Registered successfully", "success")
            return render_template("login.html", msg=f'Account created for
{username}!')
        else:
            flash("Details are invalid", "warning")
            return render_template("registration.html")
    else:
        flash("Password doesn't match", "warning")
        return render_template("registration.html")

  return render_template('registration.html')

@app.route('/userhome')
def userhome():
  return render_template('userhome.html')
```

Description:

This Python code implements a basic Flask web application that enables user registration, login, and access to a user home page. It uses MySQL for storing user data, including their name, email, password, age, and contact number. The application has several routes: the index route displays the homepage, the about route provides an about page, and the login and registration routes allow users to log in and register new accounts. During registration, the application validates the user's input, checking if the password and confirmation password match, and ensures that the user's email isn't already in use. When a user logs in, the app checks

their credentials in the database and, if valid, redirects them to a user home page. The app employs session management to temporarily store the user's email for access to personalized content. Flash messages are used to provide feedback to users during the login and registration process. While the app provides basic functionality, it has security concerns, particularly in terms of storing passwords in plain text, which should be addressed by implementing password hashing for better security.

7. 2. Summarization and Translation Module Implementation

```
# summarization_module.py
import re
import random
import nltk
from flask import Flask, render_template, request
from transformers import pipeline
from gensim.summarization.summarizer import summarize
from youtube_transcript_api import YouTubeTranscriptApi
from googletrans import Translator
from nltk.tokenize import sent_tokenize
from tensorflow.keras.models import load_model

nltk.download('punkt')

app = Flask(__name__)

# Load LSTM model
def load_lstm_model():
    print("Loading LSTM model for summarization.")
    return load_model("model.h5")

model = load_lstm_model()

def get_transcript(video_id):
    transcript_list = YouTubeTranscriptApi.get_transcript(video_id)
```

```
    transcript = ' '.join([entry['text'] for entry in transcript_list])
    words = transcript.split()
    for i in range(15, len(words), 15 + random.randint(0, 5)):
        words[i] += '.'

    return ' '.join(words)

def get_summary(text, method, percentage):
    sentences = sent_tokenize(text)
    if len(sentences) <= 1:
        return "Transcript is too short to summarize. Original Transcript: " + text

    try:
        if method == 'extractive':
            return summarize(text, ratio=float(percentage)/100)
        elif method == 'abstractive':
            summarizer = pipeline("summarization")
            summary = summarizer(text, max_length=100, min_length=5,
do_sample=False)[0]['summary_text']
            return summary
    except ValueError as ve:
        return f"Error during summarization: {ve}. Original Transcript: {text}"

def translate_summary(text, language):
    translator = Translator()
    trans_summary = translator.translate(text, dest=language)
    return trans_summary.text, trans_summary.pronunciation if trans_summary.pronunciation
else "No pronunciation available"

@app.route('/summary', methods=['GET', 'POST'])
def summary():
    if request.method == 'POST':
```

```
    url = request.form['url']
    method = request.form['method']
    percentage = request.form['percentage']
    language = request.form['language']

    video_id = re.search(r'(?:youtube\.com/watch\?v=|youtu\.be/)([a-zA-Z0-9_-]{11})', url).group(1)

    transcript = get_transcript(video_id)
    summary_text = get_summary(transcript, method, percentage)

    if transcript:
        translated_text, pronunciation = translate_summary(summary_text, language)
    else:
        summary_text = "Transcript not available."
        translated_text = ""

    return render_template('summary.html', summary=summary_text, transcript=transcript, trans_summary=translated_text)

  return render_template('summary.html')
```

Description:

This code defines a Flask web application that allows users to extract, summarize, and translate YouTube video transcripts. The app accepts a YouTube video URL, a choice between extractive or abstractive summarization, a percentage for the desired length of the summary, and a language for translation. It extracts the video transcript using the YouTubeTranscriptApi, and then summarizes the transcript either by selecting key sentences (extractive summarization using gensim) or by generating a new, concise version of the text (abstractive summarization using Hugging Face's transformers). It also translates the summary into the requested language using googletrans and displays the original transcript, the generated summary, and the translated summary in the user interface. The app loads a pre-trained LSTM model, though it's not currently being used for the summarization process. The Flask app handles the POST requests

by processing the URL, generating summaries, and rendering the results in an HTML template. While the app is functional, some improvements could be made in error handling, model usage, and the frontend interface.

7. 3. Video Processing Module Implementation

```
# video_processing_module.py
import re
import os
import cv2
import subprocess
from flask import Flask, render_template, request
from pytube import YouTube
from youtube_transcript_api import YouTubeTranscriptApi

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'videos'

def extract_video_id(url):
    match = re.search(r'(?:youtube\.com/watch\?v=|youtu\.be/)([a-zA-Z0-9_-]{11})', url)
    return match.group(1) if match else None

def get_transcript(video_id):
    transcript_list = YouTubeTranscriptApi.get_transcript(video_id)
    transcript = ' '.join([entry['text'] for entry in transcript_list])
    return transcript

@app.route('/video', methods=['POST', 'GET'])
def video():
    return render_template('video.html')

@app.route('/summary2', methods=['POST', 'GET'])
def summary2():
    if request.method == 'POST':
```

```
        url = request.form['url']
        method = request.form['method']
        percentage = request.form['percentage']

        video_id = extract_video_id(url)
        transcript = get_transcript(video_id)

        return render_template('summary2.html', transcript=transcript)

    return render_template('summary2.html')
 UPLOAD_FOLDER = 'uploads'
OUTPUT_FOLDER = 'static/outputs'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['OUTPUT_FOLDER'] = OUTPUT_FOLDER

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

if not os.path.exists(OUTPUT_FOLDER):
    os.makedirs(OUTPUT_FOLDER)

@app.route('/play', methods=['GET', 'POST'])
def play():
    if request.method == 'POST':
        file = request.files['file']
        start_time = request.form['start_time']
        end_time = request.form['end_time']
        start_ms = timestamp_to_ms(start_time)
        end_ms = timestamp_to_ms(end_time)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(filepath)
        output_file = process_video(filepath, start_ms, end_ms)
```

```
        video_url = url_for('static', filename=f'outputs/{output_file}')
        msg = "Video has processed successfully"
        path = '/static/outputs/output_video.mp4'
        return render_template('play.html', video_url=video_url, path =path, msg=msg)
    return render_template('play.html', video_url=None)

def timestamp_to_ms(timestamp):
    h, m, s = map(int, timestamp.split(':'))
    return (h * 3600 + m * 60 + s) * 1000

def process_video(input_path, start_time, end_time):
    cap = cv2.VideoCapture(input_path)
    frame_width = int(cap.get(3))
    frame_height = int(cap.get(4))
    output_file = 'output_video.mp4'
    output_path = os.path.join(app.config['OUTPUT_FOLDER'], output_file)
    out = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(*'MP4V'), 20, (frame_width,
frame_height))
    cap.set(cv2.CAP_PROP_POS_MSEC, start_time)

    while cap.get(cv2.CAP_PROP_POS_MSEC) < end_time:
        ret, frame = cap.read()
        if ret == True:
            out.write(frame)
        else:
            break

    cap.release()
    out.release()
    return output_file
if __name__ =='__main__':
    app.run(debug=True)
```

Description:

The provided code is a simple Flask web application that allows users to extract and display the transcript of a YouTube video. The application includes a route to accept YouTube video URLs, extract the video ID using a regular expression, and fetch the transcript using the YouTubeTranscriptApi. The transcript is then displayed on a separate page in the summary2.html template. The application defines a route for video processing, where it accepts POST requests with the YouTube video URL and method of summarization (though the method and percentage are not yet implemented in the logic). The code also sets up a folder for video uploads, though file handling is not currently being used. The application does not include error handling for cases where a transcript is unavailable or the video ID is incorrect. The HTML templates (video.html and summary2.html) are rendered to allow user interaction. However, some fields like method and percentage from the form are not utilized within the current code, which leaves room for future enhancements, such as implementing different methods for summarizing the transcript.

CHAPTER 8
TESTING

Testing is a crucial process in software development and quality assurance. It involves verifying that a software application or system meets specified requirements and works as expected. Here's a breakdown of the different aspects of testing:

1.Unit Testing: Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2.Integration Testing: This verifies that different components of the system work together as intended. In this case, it ensures that the generated text and images complement each other and form a coherent narrative.

3.System Testing: System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user story testing and then each component through integration testing.

4. White Box Testing: White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5. Black Box Testing: Black Box Testing is testing the software without any knowledge of

the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6. Functional Testing:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8. 1 TESTCASES

S.NO	Input	Expected Output	Test Result
1	Valid YouTube Link	Text is accurately extracted, transcribed, and summarized; summary is displayed to the user.	Pass
2	Invalid YouTube Link	System displays an error message indicating an invalid URL and prompts the user to enter a valid link.	Pass
3	YouTube Link with Non-English Audio	The model will not recognize the non-English audio	Fail
4	YouTube Link to Video with Poor Audio Quality	Text is extracted; transcription may have minor errors; summary is generated with available data without crashing.	Pass
5	Empty Input (No Link Provided)	System prompts the user to enter a valid YouTube link before proceeding.	Pass

Table 8. 1. TESTCASES

CHAPTER 9

OUTPUT SCREENS

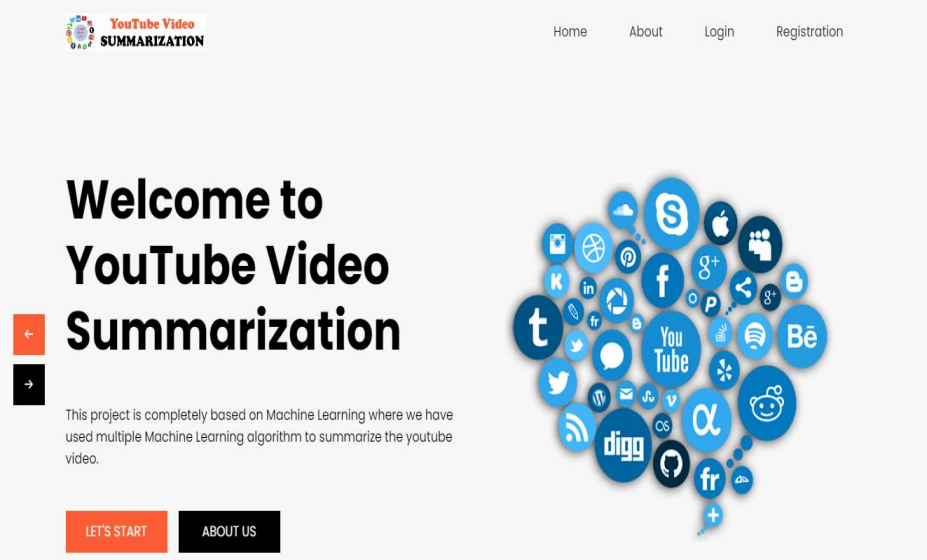

Fig 9.1. Index page

The above image 9. 1 is the user interface which a user can see after running the program module.

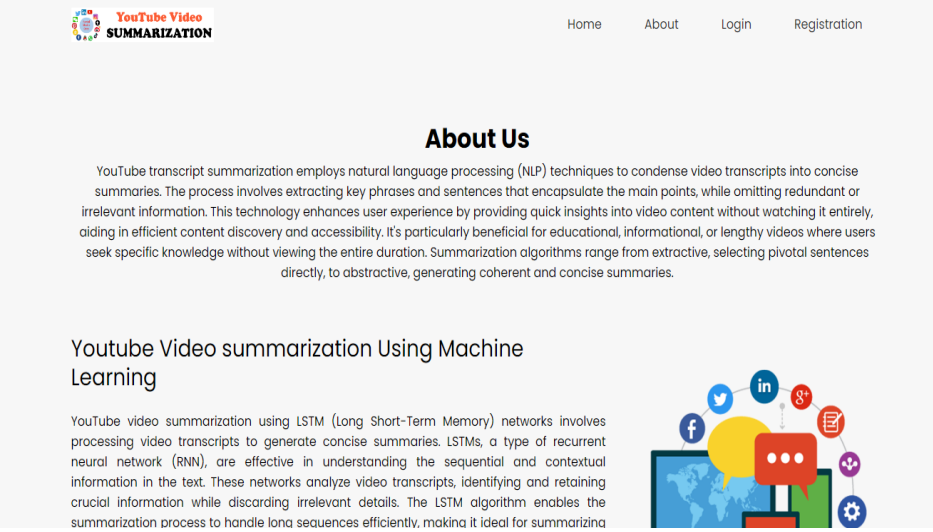

Table 9.2. About us

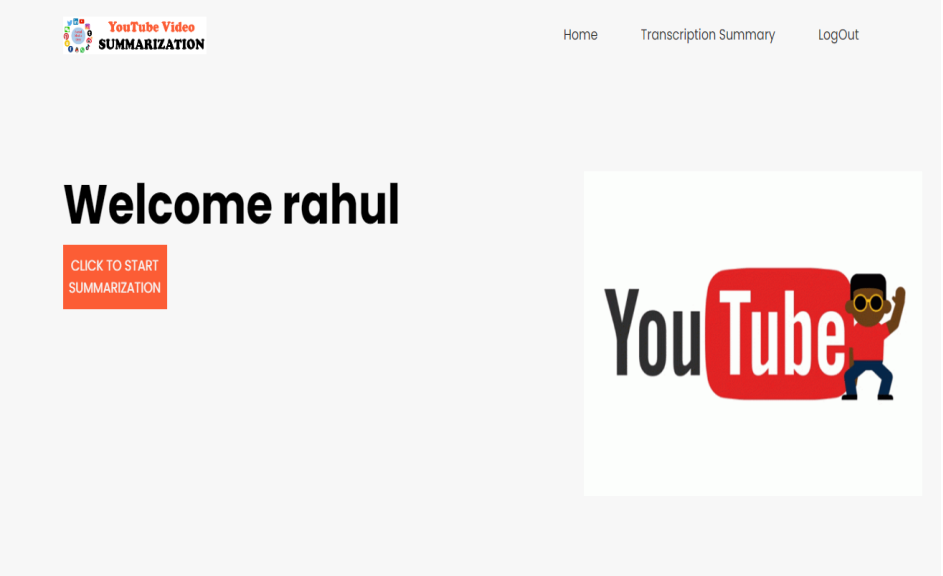

Fig 9.3. User Home page

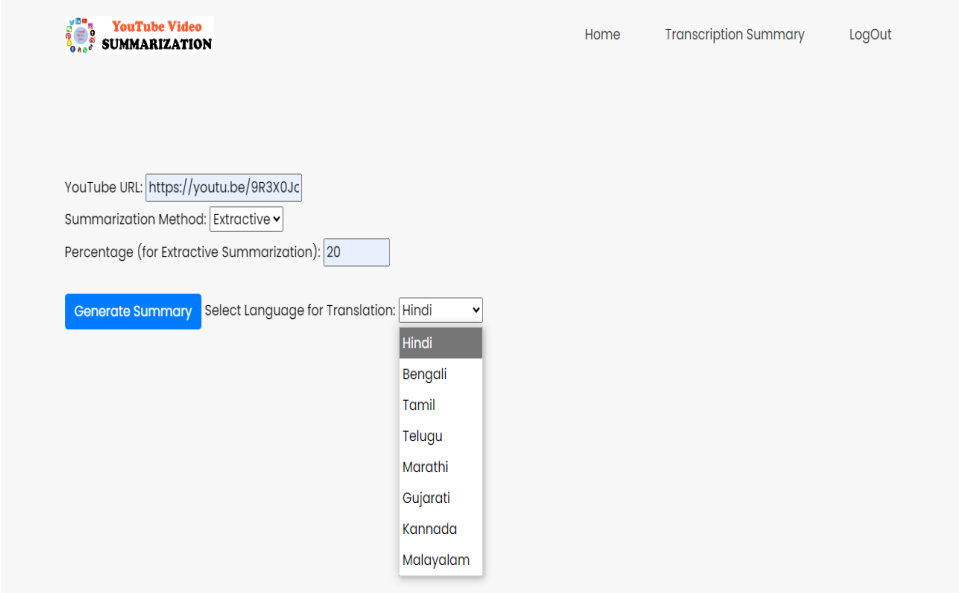

Fig 9.4. Summary Page

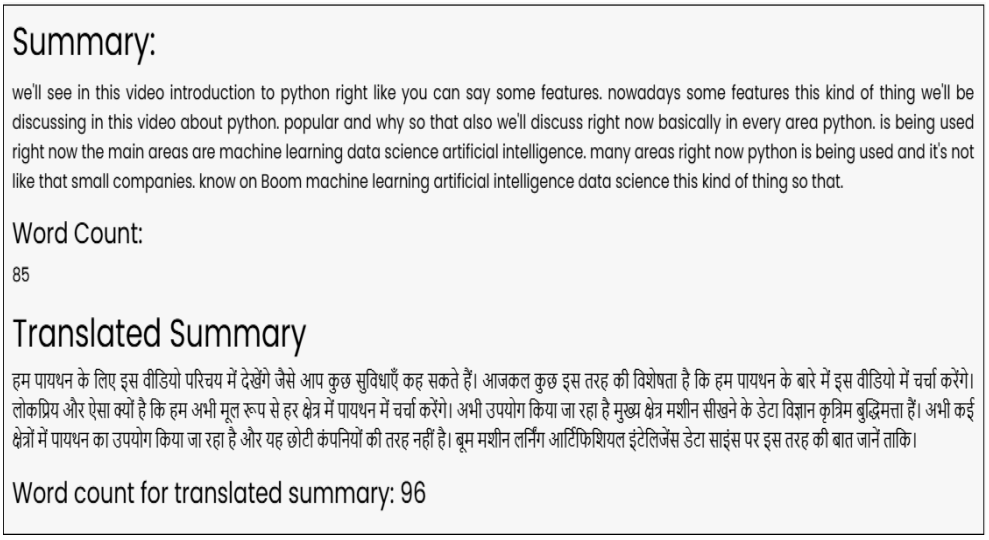

Fig 9.5. Result Page

Fig 9.6. Result Page

CHAPTER 10

DEPLOYMENT

The below is the step by step guide to deploy the Project:

I. INSTALL PYTHON

- Download and install Python from official Python website.
- Ensure Python is added to your system's PATH during installation.
- Optionally, install Anaconda for managing Python packages and environments easily.

II. Install XAMPP:

1. Download XAMPP:

- Visit the official XAMPP website: https://www.apachefriends.org/index.html.
- Choose the appropriate version for your operating system (Windows, Linux, or macOS).
- Click the "Download" button to begin downloading the XAMPP installer.

2. Install XAMPP on Windows:

- **Run the Installer:**
  - Once the XAMPP installer file is downloaded, double-click on it to run.
- **Choose Installation Language:**
  - Select the language (usually English) and click "OK."
- **Select Components to Install:**
  - The installer will ask you to select the components to install. By default, Apache, MySQL, PHP, phpMyAdmin, and others are selected. You can leave them selected if you want the full stack. Click Next.
- **Choose Installation Directory:**
  - Choose the directory where you want to install XAMPP. The default is C:\xampp. Click Next.
- **Start Menu Folder:**
  - Choose a folder in the Start Menu for XAMPP shortcuts. You can leave this as default or change it. Click Next.

- **Ready to Install:**
  - o Click Install to start the installation process. This might take a few minutes.

3. Launch XAMPP:

- **Start the XAMPP Control Panel:**
  - Once the installation is finished, you can launch the XAMPP Control Panel.
  - The installer will ask if you want to start the XAMPP Control Panel immediately. Select Finish.
- **Start MySQL:**
  - Open the XAMPP Control Panel.
  - In the Control Panel, click Start next to MySQL to start the database server.
  - After MySQL is started, it's status should change to "Running" with green indicators.

4. Test the Installation:

- **Check the XAMPP Dashboard:**
  - Open your web browser and type http://localhost/ in the address bar.
  - If everything is installed correctly, you should see the XAMPP dashboard, which confirms that your server is up and running.

III. EXECUTION

In terminal type:

Step-1) cmd

Step-2) conda activate python3.10.8

Step-3) python app.py

CHAPTER 11
PERFORMANCE EVALUATION

The performance of the YouTube video transcript summarization system is evaluated based on accuracy, efficiency, scalability, and user experience. The system employs both extractive and abstractive summarization techniques, leveraging an LSTM-based model for extractive summarization and a transformer-based model for abstractive summarization. The accuracy of the generated summaries depends on the quality of the extracted transcript, with cleaner transcripts leading to better summarization results. The extractive approach is effective for capturing key sentences but may sometimes fail to provide a coherent flow of information, whereas the abstractive approach generates more natural summaries but may introduce minor distortions in meaning.

In terms of efficiency, the system processes transcripts in a reasonable time, though abstractive summarization is computationally more expensive due to the transformer-based model. The system's ability to handle long transcripts is influenced by memory and processing constraints, which can be optimized by reducing unnecessary computations and improving model efficiency. Additionally, the integration of the Google Translate API allows for multilingual support, making the system accessible to a wider audience, though translation accuracy varies depending on the target language and linguistic complexity.

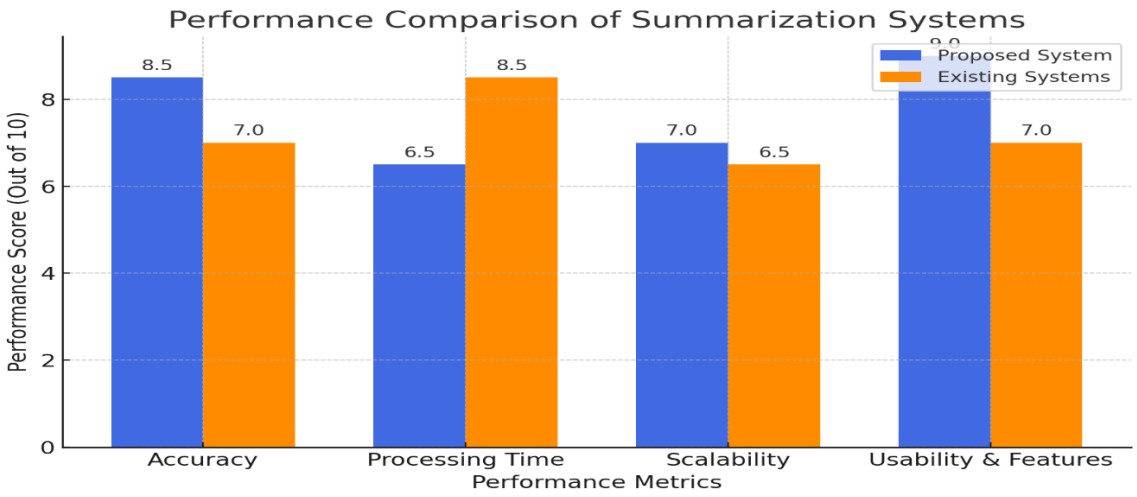

Fig 11.1. Performance Evaluation

1. Accuracy of Summarization: The system provides both extractive and abstractive summaries. Extractive summarization uses the LSTM model, ensuring key points are retained, while abstractive summarization leverages transformer models for natural language generation. The quality of summaries is evaluated using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores to measure similarity with human-generated summaries.

2. Processing Time: The efficiency of the system is measured by the time taken to generate summaries. Extractive summarization is generally faster due to direct text extraction, while abstractive summarization takes longer due to deep learning computations.

3. Scalability & Load Handling: The system is tested under different workloads, from short videos to long lectures, ensuring database optimization and parallel processing for handling multiple requests efficiently.

4. Accuracy of Transcript Extraction: The YouTube Transcript API is used to extract transcripts, and its accuracy depends on video captions. Testing is done across different languages and accents to measure transcription correctness.

5. User Experience & Language Translation: The translation feature is evaluated based on semantic accuracy and user feedback to ensure meaningful translations without loss of context. Performance is enhanced using Google Translate API for multilingual support.

CHAPTER 12

COMPARISION WITH EXISTING SYSTEM

1. **Multiple Summarization Methods:**
    - The user can choose between **extractive** and **abstractive summarization**, offering more flexibility compared to many existing systems that provide only one method.
2. **Customizable Summary Length**:
    - The ability to specify the percentage of the summary's length allows users to control how concise or detailed the summary should be.
3. **Multilingual Support**:
    - Translates the summary into various languages using **Google Translate**.
    - Provides pronunciation for the translated text, making it useful for non-native speakers.
4. **Transcript Formatting**:
    - Adds punctuation and splits transcripts into more readable sentences, improving the readability of raw YouTube video transcripts.
5. **Simple Web Interface**:
    - The system is easy to use, with simple forms for entering video URLs and selecting summarization options. This user-friendly interface makes the system accessible to non-technical users.
6. **Database Integration**:
    - Allows for user registration and login, providing a personalized experience for users (saving preferences, viewing history, etc.).

CHAPTER 13
CONCLUSION

This project effectively tackles the challenge of distilling essential information from lengthy YouTube videos by automating the summarization process. Utilizing advanced machine learning techniques, specifically an LSTM-based model, combined with natural language processing and speech recognition, the system transforms spoken content into concise, coherent summaries. The methodology ensures that users receive accurate and relevant summaries by carefully processing each step—from audio extraction and transcription to preprocessing and summarization.

The inclusion of a translation feature enhances the system's accessibility, allowing non-English speakers to benefit from the summaries in their native languages via Google Translator. This broadens the tool's usability across different demographics and regions. The user-friendly interface ensures that individuals with varying technical skills can easily input video links and access the generated summaries.

By saving users time and effort in consuming video content, the project offers significant value in educational, professional, and personal contexts. It demonstrates the practical application of AI and machine learning technologies in addressing real-world problems. Future enhancements could involve integrating more sophisticated models for improved summarization accuracy and expanding language support for both transcription and translation. Overall, the project stands as a testament to how technology can streamline information consumption in the digital age.

CHAPTER 14
FUTURE ENHANCEMENTS

Integration of Advanced Summarization Models:

Transformer-Based Models: Upgrade the summarization component by integrating transformer-based models like BERT, GPT-3, or T5. These models have shown superior performance in natural language processing tasks and can provide more accurate and coherent summaries.

Customization of Summary Length and Style: Allow users to adjust the length of the summary or choose the summarization style (e.g., bullet points, narrative). This provides flexibility to meet different user needs.

Personalization and User Preferences:

User Profiles and Settings: Allow users to create profiles where they can set preferences such as default language, summary length, or preferred summarization style.

Machine Learning Personalization: Implement algorithms that learn from user interactions to provide more personalized summaries over time.

Enhanced Speech Recognition:

Multilingual Speech Recognition: Improve the system's ability to transcribe audio in multiple languages by integrating advanced speech recognition APIs like Google Cloud Speech-to-Text or DeepSpeech. This would broaden the user base and applicability of the tool.

Noise Reduction and Audio Enhancement: Implement preprocessing steps to clean and enhance the audio before transcription, improving accuracy, especially for videos with poor audio quality.

CHAPTER 15
REFERENCES

[1] Kiran K N , “Video Summarization using NLP,” IJRET,2021.
IRJET-V8I8411.pdf

[2] Prof. Monali Bansode,“Extractive Text and Video Summarization using TF-IDF Algorithm ”,IJRASET,2022.
Extractive Text and Video Summarization using TF-IDF Algorithm

[3] Sarah S. Alrumiah,“Educational Videos Subtitles’ Summarization Using Latent Dirichlet Allocation and Length Enhancement ”,CMC, 2022.
https://www.researchgate.net/publication/355194117_Educational_Videos_Subtitles'_Summarization_Using_Latent_Dirichlet_Allocation_and_Length_Enhancement

[4] Shraddha Yadav, “Summary And Keyboard Extraction From Youtube Video Transcript”,IJRMETS,2021.
1628083490.pdf

[5] Madhu S.Nair,“Static video summarization using multi-CNN with sparse autoencoder and random forest classifier”, IEEE,2020.
Static video summarization using multi-CNN with sparse autoencoder and random forest classifier | Signal, Image and Video Processing

[6] Snehal Hiray, “Video Summarization ”,IJRET, 2017.
Video Summarization by IRJET Journal - Issuu

[7] Prof. Tanuja Sachin Khatavkar, “Video Summarization using NLP”,IJSDR,2023.
IJSDR2306069.pdf

[8] Surabhi Adhikari,”NLP Based Machine Learning Approaches for Text Summarization”,IEEE,2020.
NLP based Machine Learning Approaches for Text Summarization | IEEE Conference Publication | IEEE Xplore

[9] YoungJin Suh,” Deep Learning-based Video Summarization”,IJASEIT, 2021.
(PDF) Deep Learning-based Video Summarization