VIDEO SUMMARIZER USING NATURAL LANGUAGE PROCESSING

A PROJECT REPORT

Submitted by

KAVIN KIRTHIK RP	**811722104071**
NIKHAAL AHAMED A K	**811722104101**
RAGHUL P	**811722104116**

in partial fulfillment of the requirements for the award degree of

Bachelor in Engineering

20CS7503 DESIGN PROJECT - 3

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM - 621112

NOVEMBER 2025

VIDEO SUMMARIZER USING NATURAL LANGUAGE PROCESSING

A PROJECT REPORT

Submitted by

KAVIN KIRTHIK RP	**811722104071**
NIKHAAL AHAMED A K	**811722104101**
RAGHUL P	**811722104116**

in partial fulfillment of the requirements for the award degree of

Bachelor in Engineering

20CS7503 DESIGN PROJECT - 3

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM - 621112

NOVEMBER 2025

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM - 621112

BONAFIDE CERTIFICATE

The work embodied in the present project report entitled **VIDEO SUMMARIZER USING NATURAL LANGUAGE PROCESSING** has been carried out by the students **KAVIN KIRTHIK RP, NIKHAAL AHAMED A K, RAGHUL P**, the work reported herein is original and we declare that the project is their own work, except where specifically acknowledged, and has not been copied from other sources or been previously submitted for assessment.

Date of Viva Voce: ……………………

Ms. S. UMA MAGESHWARI, M.E.,	**Mr. R. RAJAVARMAN,M.E.,(Ph.D.,)**
SUPERVISOR	HEAD OF THE DEPARTMENT
Assistant Professor	Assistant Professor (Sr. Grade)
Department of CSE	Department of CSE
K. Ramakrishnan College of Technology	K. Ramakrishnan College of Technology
Samayapuram -621 112	Samayapuram -621 112

INTERNAL EXAMINER **EXTERNAL EXAMNIER**

ABSTRACT

Offline video summarization is achieved through an integrated system that combines Whisper-based transcription with Ollama-driven abstractive text generation. Video audio is extracted, converted into accurate transcripts, and transformed into concise summaries without relying on cloud services. A Unity interface coordinates the workflow, enabling users to load videos, initiate processing, and access the final output effortlessly. Operating entirely on local hardware ensures privacy, reliability, and consistent performance. The approach offers an effective way to understand lengthy video content quickly, supporting students, educators, and professionals who require fast retrieval of key information from long-form recording.

Keywords:

Video Summarization, Whisper ASR, Ollama LLM, Offline Processing, Audio Extraction, NLP, Unity Interface, Transcript Generation.

ACKNOWLEDGEMENT

We thank our **Dr. N. Vasudevan**, Principal, for his valuable suggestions and support during the course of my research work.

We thank our **Mr. R. Rajavarman,** Head of the Department, Assistant Professor (Sr. Grade), Computer Science and Engineering, for his/her valuable suggestions and support during the course of my research work.

We wish to record my deep sense of gratitude and profound thanks to my Guide **Ms. S. Uma Mageshwari**, Assistant Professor, Computer Science and Engineering for her keen interest, inspiring guidance, constant encouragement with my work during all stages, to bring this thesis into fruition.

We are extremely indebted to our project coordinator **Mr. M. Saravanan,** Assistant Professor, Computer Science and Engineering , for his valuable suggestions and support during the course of my research work.

We also thank the faculty and non-teaching staff members of the Computer Science and Engineering, K. Ramakrishnan College of Technology (Autonomous), Samayapuram , for their valuable support throughout the course of my research work.

Finally, we thank our parents, friends and our well wishes for their kind support.

SIGNATURE

____

____

____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ASR	-	Automatic Speech Recognition
LLM	-	Large Language Model
NLP	-	Natural Language Processing
IPC	-	Inter Process Communication
FPS	-	Frames Per Second
API	-	Application Programming Interface
HCI	-	Human–Computer Interaction
GUI	-	Graphical User Interface
FFmpeg	-	Fast Forward Moving Picture Experts Group
ML	-	Machine Learning
AI	-	Artificial Intelligence
UI	-	User Interface
CPU	-	Central Processing Unit
RAM	-	Random Access Memory

CHAPTER – 1

INTRODUCTION

1.1 DESCRIPTION

Video summarization is the process of condensing long videos into shorter versions while keeping the important information intact. It has evolved significantly over the years, progressing from simple manual editing to advanced automated solutions created through artificial intelligence. Traditional summarization required users to rewatch entire videos, manually select key moments, and rewrite the important details. This process was slow, labor intensive, and often inaccurate. With recent advancements in machine learning, summarization algorithms can now analyze the structure of speech, detect contextual meaning, and automatically generate summaries. This makes information consumption easier, especially when dealing with long educational, professional, or technical videos.

1.2 OVERVIEW OF MACHINE LEARNING BASED SUMMARIZATION

Machine learning based summarization goes beyond simple text shortening. It uses advanced models capable of understanding context, meaning, and the relationship between sentences. Speech to text models like Whisper convert spoken language into readable text with high precision, even when the audio contains noise, interruptions, or unclear speech. Large language models, such as the ones used through Ollama, analyze long transcripts, detect recurring themes, and produce summaries that are structured, meaningful, and easy to understand. These models are trained on large datasets, enabling them to recognize patterns, identify important information, and generate summaries that reflect the core message of the original video. Machine learning enables a level of accuracy and understanding that traditional rule based methods cannot achieve.

1.2.1 NEED FOR AUTOMATED SUMMARIES

The need for automated video summarization has become more urgent due to the exponential growth of digital video content. Students often face hour long lectures and tutorials that require repeated revision. Professionals attend recorded meetings and training sessions that they rarely have time to review fully. Researchers handle conference presentations, expert talks, and interviews that must be analyzed quickly. Manual summarization is impractical in all these scenarios. Automated summarization helps users save time, improve productivity, reduce workload, and access relevant information without watching the entire video. It also provides better clarity, consistency, and accuracy compared to manual efforts.

1.2.2 APPLICATIONS OF VIDEO SUMMARIES

Video summaries have applications across many fields. In education, they help students revise lectures, understand topics faster, and extract important points during exam preparation. In corporate sectors, employees use summaries to review meetings, project discussions, and training sessions. In media and content creation, video summaries help writers, editors, and creators extract scripts, captions, and documentation. In research, scholars can quickly understand academic presentations, interviews, and domain specific discussions. Overall, video summarization supports quick knowledge access, better decision making, and improved workflow efficiency.

1.3 OFFLINE VIDEO SUMMARIZATION SYSTEM USING OLLAMA AND UNITY

This project introduces an offline system where all processing happens on the user’s computer without requiring internet connectivity. Unity serves as the application environment that allows users to input videos easily. The Whisper model is run locally to convert speech into text, ensuring privacy and eliminating dependency on cloud servers. The transcript generated by Whisper is then processed using Ollama, which provides a local large language model environment that performs summarization efficiently. The offline approach ensures that the user's data remains secure, summarization works even without internet access, and performance is consistent. This system is suitable for environments where privacy, speed, and reliability are important.

1.4 SYSTEM ARCHITECTURE

The system architecture integrates all core components required to transform video input into a summarized text output. Each module performs a dedicated task, and together they form a seamless workflow. The architecture includes a Unity front end interface for user interaction, an audio extraction module for isolating audio from video, a Whisper engine for transcription, and an Ollama based summarization model for generating the final summary. All these components operate offline, ensuring smooth performance without external dependencies.

1.4.1 UNITY FRONT END INTERFACE

The Unity interface serves as the user friendly platform where users upload videos or provide video URLs. It controls the entire sequence of processing steps and displays the final summarized text. Unity allows easy integration of external models and provides a visually clear environment for users who may not have technical expertise.

1.4.2 AUDIO EXTRACTION MODULE

This module extracts the audio track from the input video. It ensures that the audio is converted into a format suitable for Whisper. Proper extraction is important because the accuracy of the transcript depends on the clarity of the audio file being processed.

1.4.3 WHISPER TRANSCRIPTION ENGINE

Whisper receives the extracted audio and converts it into text. It uses deep learning techniques to recognize speech patterns and produce accurate transcripts. The transcription includes spoken words, technical terms, and detailed phrases, all of which are essential for a high quality summary.

1.4.4 OLLAMA LLM SUMMARIZER (3B QUANTIZED MODEL)

The summarizer uses a quantized 3B language model running in Ollama. Quantization reduces memory usage and increases speed while maintaining summarization quality. The model analyzes long transcripts, identifies important segments, and generates a clear summary that captures the essential meaning of the video.

1.4.5 OFFLINE PROCESSING WORKFLOW

All operations, from video input to final summary generation, occur locally on the user's machine. This reduces delays, improves privacy, and ensures continuous functionality even without internet access. The offline workflow enhances reliability and makes the system suitable for educational institutions, private organizations, and individuals who require secure processing.

1.5 SPEECH TO TEXT PROCESSING USING WHISPER

Whisper is a modern speech recognition model capable of transcribing spoken content into text with high accuracy. It is designed to handle different accents, speech speeds, and background conditions. Whisper processes the extracted audio from the video and generates a detailed transcript that captures all spoken information. This transcript becomes the foundation for further summarization. Because Whisper is open source and can run completely offline, it is suitable for sensitive or private video content.

CHAPTER – 2

LITERATURE SURVEY

2.1 ASOVS: ABSTRACTIVE SUMMARIZATION OF VIDEO SEQUENCES

Aniqa Dilawari and Muhammad Usman Ghani Khan use abstractive video summarization focuses on generating textual descriptions that represent the deeper meaning, context, and intent of a video, rather than merely extracting key frames or selecting existing sentences. The goal is not only to shorten the video but also to transform its visual content into a coherent narrative that resembles human written summaries. In this context, the ASoVS model introduces a deep learning-based solution that leverages neural network architectures specifically designed to interpret video sequences and convert them into meaningful language outputs. The system begins by extracting visual features from individual video frames, often using convolutional neural networks to capture spatial information such as objects, scenes, actions, and motion patterns.

These extracted features are then fed into sequence-to-sequence models, which process temporal relationships across frames to understand how events unfold over time. By learning both spatial and temporal dependencies, the model develops a comprehensive understanding of the video's content. Unlike extractive summarization methods, which simply pick representative frames or sentences, ASoVS aims to rephrase and reconstruct the video’s meaning using natural language generation techniques. This allows the generated summaries to be more fluent, descriptive, and semantically rich, closely resembling the way a human observer would describe a video after watching it. The model is trained on datasets containing paired video segments and corresponding textual descriptions, enabling it to learn how visual cues relate to linguistic expressions.

2.2 CYCLE SUM: CYCLE CONSISTENT ADVERSARIAL LSTM NETWORKS FOR UNSUPERVISED VIDEO SUMMARIZATION.

Li Yuan, Francis EH Tay, Ping Li, Li Zhou, Jiashi Feng uses Cycle SUM framework introduces an unsupervised approach to video summarization that eliminates the need for manually annotated training data. Traditional supervised video summarization relies on human labeled datasets, where annotators identify important frames or segments. However, such datasets are expensive to create, limited in size, and often fail to generalize across different types of videos. Cycle SUM addresses this limitation by employing a cycle consistent adversarial learning strategy that allows the model to learn meaningful summaries directly from unlabeled videos. The architecture consists of two major components: a frame selector and a cycle consistent evaluator.

The frame selector is built using a bidirectional LSTM network, which processes video frames in both forward and backward directions. This bidirectional structure allows the selector to capture long range temporal dependencies and understand how scenes evolve across the entire video. The selector outputs a sequence of importance scores that determine which frames should be included in the summarized video. Its goal is to produce a condensed version of the video that retains all essential information while removing redundant or irrelevant content. The cycle consistent evaluator, which forms the core of the Cycle SUM approach, consists of two generative adversarial networks. The forward GAN reconstructs the original video from the generated summary, while the backward GAN reconstructs the summary from the original video. By enforcing consistency between these outputs, the model ensures that the summary contains enough information to faithfully reproduce the original sequence.

This cycle consistency constraint encourages the selector to preserve critical content from the input video, making the summary both compact and informative.

2.3 DEEP REINFORCEMENT LEARNING FOR UNSUPERVISED VIDEO SUMMARIZATION

Kaigang Zhou utilize deep reinforcement learning has emerged as a highly effective approach for handling complex decision-making problems, and its application to unsupervised video summarization demonstrates its potential for generating high quality summaries without the need for manually annotated datasets. In this study, the authors introduce a reinforcement learning framework that treats video summarization as a sequential decision-making task, where the model must determine which frames to include in the final summary. Instead of relying on fixed rules or predetermined labels, the system learns through experience by interacting with a reward driven environment that guides its choices. The model uses a Bidirectional Recurrent Neural Network to process video frames and estimate their importance. By analyzing the sequence both forward and backward, the network captures temporal dependencies and contextual relationships between frames, allowing it to identify segments that contribute most to the narrative or informational structure of the video.

Reinforcement learning is then applied to refine the selection process. The model receives reward signals based on how well its chosen frames represent the original video while avoiding redundancy. The reward function is carefully designed to balance two key objectives: diversity, which ensures the summary includes different types of content, and representativeness, which guarantees that critical events and scenes are preserved. One of the significant strengths of this approach is its ability to operate entirely without human annotations. Instead of learning from labelled datasets, the model improves its summarization strategy through continuous feedback, allowing it to adapt and generalize to different types of videos. The learning process encourages the model to explore various combinations of frame selections and gradually converge toward an optimal summarization policy.

2.4 EXTRACTIVE AUTOMATIC TEXT SUMMARIZATION USING SPACY

Swaranjali Jugran uses approach makes use of SpaCy, a widely adopted natural language processing library developed in Python, to perform extractive summarization by analyzing the linguistic structure of text in depth. SpaCy provides a comprehensive set of tools, including tokenization, part of speech tagging, dependency parsing, and named entity recognition, which together enable the system to understand how words and sentences are formed and how they relate to each other within a document. Using these capabilities, the summarization method evaluates each sentence according to its linguistic significance, the presence of important terms, and its contextual contribution to the overall meaning of the text. Sentences that score higher in these aspects are considered informative and are selected to form the final summary. Although the technique does not engage in rewriting or generating new sentences, it is able to consistently capture the essential portions of the content by directly extracting the most meaningful segments.

This approach is particularly effective for small and medium sized datasets, where the objective is to obtain quick and reliable summaries without excessive computational load. The method demonstrates how rule based natural language processing, when supported by an efficient and modern library like SpaCy, can still produce high–quality results even in a landscape dominated by deep learning models. It highlights the practicality of extractive summarization in situations where speed, interpretability, and resource efficiency are important. The findings reinforce the idea that extractive summarization continues to play a valuable role in text processing, especially for applications where maintaining the original wording is desirable.

SpaCy based summarization is able to preserve grammatical correctness, retain core information, and present users with concise content that reflects the intent of the original material.

2.5 FULLY CONVOLUTIONAL SEQUENCE NETWORKS FOR VIDEO SUMMARIZATION

Mrigank Rochan used the concept of Fully Convolutional Sequence Networks, commonly referred to as FCSN, represents a significant step forward in the field of video summarization. Earlier summarization methods relied heavily on recurrent neural network models such as LSTMs or GRUs, which process video frames sequentially. While effective in learning temporal patterns, recurrent models often encounter limitations when handling long video sequences. They require more time to train, struggle with vanishing gradients, and process frames one step at a time, which slows down overall computation. These drawbacks make traditional recurrent approaches less suitable for modern large scale video datasets. FCSN addresses these limitations by replacing recurrent layers with temporal convolutional layers that are capable of analyzing sequences in parallel. Instead of processing each frame one after another, the network applies convolution operations across time, allowing multiple frames to be examined simultaneously.

This parallel processing design greatly improves the speed of training and reduces computational overhead. Temporal convolution also helps the model capture both short term and long term dependencies by stacking multiple convolutional layers with different receptive fields. As a result, the model can effectively understand the structure of events occurring across the entire video. The main task of the FCSN architecture is to predict the importance score of each video segment. These scores indicate how relevant or informative each segment is, and the system uses them to select keyframes or key shots that best represent the content of the original video. By identifying these important intervals, the network produces concise summaries that maintain the essential storyline or informational structure of the input video. Through experimental evaluation, the authors demonstrate that FCSN achieves better performance compared to LSTM based models in terms of both accuracy and computational efficiency.

2.6 HIERARCHICAL RECURRENT NEURAL NETWORK FOR VIDEO SUMMARIZATION

Bin Zhao study introduces a hierarchical recurrent neural network architecture designed specifically to address challenges in long video summarization. Traditional recurrent neural networks, including LSTM based models, are effective at learning temporal dependencies but often struggle with long input sequences, such as full length videos. These models are typically able to process only short clips efficiently due to limitations in memory handling, vanishing gradients, and computational cost. As a result, summarizing long videos using conventional RNN structures often leads to incomplete coverage of important content or loss of temporal continuity. To address these limitations, the authors propose a hierarchical recurrent neural network, referred to as HRNN, which processes video content at multiple levels rather than treating the video as one continuous sequence. The architecture separates the summarization process into two layers: a lower level RNN and a higher level RNN. The lower level RNN deals with shorter video subshots extracted from the original sequence. By processing smaller, manageable chunks of frames, this layer is able to capture detailed temporal dependencies within short intervals without losing context.

The output of the first layer then becomes the input for the higher level RNN, which analyzes the combined representation of subshots and determines which ones should be included in the final summary. This hierarchical design allows the network to capture long range temporal structure and short term motion dynamics simultaneously. The model is trained to identify key subshots that best represent the overall video content. By encod-ing the importance of each subshot in a structured manner, the system generates summaries that maintain logical flow, meaningful transitions, and visual diversity. Compared to traditional RNN based summarization models, H RNN demonstrates superior performance in capturing both fine grained and global temporal relationships. This leads to more coherent and informative summaries, particularly for long videos that extend beyond the capacity of standard sequential models.

2.7 METHOD OF TEXT SUMMARIZATION USING LSA

Hritvik Gupta used text summarization has undergone a major transformation in recent years as the volume of digital information continues to grow at an unprecedented rate. Users now expect immediate access to concise and meaningful content, making the development of efficient summarization techniques more important than ever. The referenced study presents a hybrid summarization framework that integrates traditional statistical approaches with advanced deep learning based contextual models to address the limitations of earlier systems. Classical techniques such as Latent Semantic Analysis are used to capture semantic structures within text documents by examining patterns of word co occurrence, while TF IDF weighting helps identify the most significant and frequently referenced terms.

These methods provide a strong foundation for ranking sentences based on statistical importance. However, one major weakness of purely statistical systems is their inability to understand deeper contextual relationships, such as sentence meaning, intent, tone, and nuance. To overcome these limitations, the authors incorporate BERT, a transformer based language model known for its ability to interpret text bidirectionally. BERT processes a sentence by analyzing the words before and after it, allowing it to understand context more accurately than earlier unidirectional models. By merging the semantic capabilities of LSA and TF IDF with the contextual depth provided by BERT, the hybrid model is able to evaluate text from multiple perspectives. This combination enables the system to select sentences that are statistically important while also being contextually relevant to the overall narrative. The experimental results demonstrate clear improvements in summary coherence, sentence organization, and information coverage. The hybrid approach significantly reduces redundancy by avoiding repeated or overlapping content, and it enhances readability by selecting sentences that align more closely with human reasoning.

2.8. UNSUPERVISED VIDEO SUMMARIZATION WITH ADVERSARIAL LSTM NETWORKS (SUM GAN)

Behrooz Mahasseni used unsupervised Video Summarization with Adversarial LSTM Networks (SUMGAN) represents a major advancement in the field of automated video summarization by introducing generative adversarial learning into the summarization pipeline. Traditional supervised methods required large amounts of annotated training data, where human reviewers manually selected important video frames or segments. Such annotation processes were costly, time consuming, and limited in scalability.

SUM GAN addresses this challenge by proposing a completely unsupervised system that learns to summarize videos without any labeled examples, making it far more practical for real world applications involving large video collections. The SUM GAN framework consists of two main components: a generator and a discriminator. The generator is implemented using an LSTM network that processes video frames sequentially and predicts which frames should be included in the summarized version. Its goal is to produce a shorter sequence that captures the essential structure, flow, and content of the original video. The discriminator, on the other hand, evaluates the output produced by the generator. It attempts to distinguish between real video sequences and the generated summaries. By training these two networks in opposition to each other, the model gradually improves its ability to produce summaries that are both compact and representative of the full video. This adversarial training setup ensures that the generated summaries maintain important characteristics of the video, such as diversity of content, temporal continuity, and relevance of key scenes. At the same time, the model learns to remove redundant or unimportant segments, effectively reducing the video length while preserving essential information. The unsupervised nature of the system makes it especially valuable for large data environments, where manually labeling thousands of videos is impractical. The results show that adversarial learning is capable of producing summaries that are competitive with, and in some cases superior to, those generated by supervised models.

2.9 YOUTUBE VIDEO SUMMARIZER USING NLP

Yogendra Singh, Rishu Kumar concentrated the increasing relevance of natural language processing techniques in the domain of YouTube video summarization. As online video platforms grow exponentially, the amount of content uploaded each minute has created challenges for users who need quick access to meaningful information without watching entire videos.The authors highlight that YouTube, being one of the largest and most rapidly expanding video platforms, contains vast quantities of educational, entertainment, and informational content, making efficient summarization tools essential for improving accessibility and user experience.

The study explains how NLP can be used to extract and analyze textual elements associated with videos, such as transcripts, captions, user comments, title descriptions, and metadata. By processing these elements, NLP based systems can identify important segments, detect topics, and generate summaries that reflect the core meaning of the original content. The review categorizes different NLP techniques used for summarization, including classical approaches such as text ranking, frequency based extraction, and topic modeling, as well as advanced deep learning methods involving transformers, sequence to sequence models, and attention mechanisms. These techniques enable the system to understand contextual relationships, semantic meaning, sentiment, and user intent. One of the key contributions of this review is the identification of challenges in video summarization research, such as noisy transcripts, multilingual content, varying video genres, and the lack of standardized benchmark datasets for fair performance comparison.

The authors emphasize that despite such challenges, NLP remains promising for extracting useful information from large volumes of video data efficiently.

2.10 YOUTUBE TRANSCRIPT SUMMARIZER

Vijaya Babu Panthagani, Vijaya Deepika Reddy Duggempudi concentrated on transcript based summarization system designed to extract meaningful information from YouTube videos by leveraging natural language processing and machine learning techniques. As the volume of online video content continues to grow rapidly, users increasingly rely on platforms like YouTube for educational material, entertainment, tutorials, and domain specific information. However, most videos contain repetitive explanations, unnecessary dialogue, promotional segments, or irrelevant content that makes it difficult for viewers to quickly identify important points.

The authors address this challenge by developing a system that automatically summarizes YouTube video transcripts, enabling users to obtain essential information in a shorter and more manageable format. The core idea of this work is to generalize long, unstructured video transcripts into coherent abstractive summaries without losing key informational elements. The system begins by extracting transcripts made available through YouTube's captioning service or through automatic speech recognition. These transcripts often contain noisy data, filler words, informal speech patterns, and long, unstructured passages.

The authors apply multiple NLP preprocessing steps to clean and segment the text, ensuring that only relevant and meaningful sentences proceed to the summarization stage. The summarizer uses a combination of machine learning and natural language processing algorithms to convert lengthy transcripts into shorter narratives that preserve essential content. Abstractive summarization plays a central role in this system. Instead of simply selecting important sentences, the model rephrases and reorganizes ideas to produce a concise and fluent summary. This allows the final output to read more naturally and better reflect the original intent of the video, while removing redundant or irrelevant segments.

CHAPTER – 3
EXISTING SYSTEM

The existing systems currently available for video summarization rely heavily on conventional natural language processing pipelines and cloud based transcription services, both of which present several limitations. In most cases, the summarization workflow begins by taking a video input from platforms such as YouTube and attempting to obtain textual data from it. Instead of directly processing the raw audio from the video, these systems depend on auto generated transcripts provided by YouTube or manually uploaded subtitle files. Such transcripts are often incomplete or inaccurate because they are generated through automated captioning mechanisms that frequently struggle with fast speech, overlapping dialogue, background disturbances, and varied accents. As a result, important information may be missed, and the overall context of the conversation may not be captured effectively. Since these systems rarely analyze the audio stream at the source, they lack the ability to handle variations in tone, emphasis, or speaker intent, which significantly limits the quality of the final summaries.

Once the transcript is obtained, existing systems generally apply traditional NLP methods to extract information. These techniques focus on basic lexical and statistical features such as word frequency, sentence length, keyword density, and simple syntactic structures. While such approaches can successfully highlight frequently occurring terms or detect sentences with high information weight, they remain shallow in understanding the deeper meaning of the content. They do not possess the ability to interpret semantic relationships between sentences, understand the flow of ideas, or capture subtle contextual transitions that naturally occur in spoken communication. As a result, the summarization produced tends to be extractive in nature, selecting isolated sentences rather than generating an integrated and coherent summary. These extractive summaries often lack narrative flow, may include redundant information, and sometimes fail to convey the original intent of the speaker, especially in complex instructional or technical videos.

Another major limitation of existing systems is their dependence on online services. Most current tools require cloud based APIs for transcription, text preprocessing, and even summarization. This dependency creates several drawbacks, including privacy concerns when sensitive audio or video data is transmitted to third party servers. There are also practical issues like network instability, restricted access in offline or low connectivity environments, and the recurring costs associated with subscription based API usage. For many users, such constraints make these systems inaccessible or impractical, especially when dealing with confidential content or when needing to process large volumes of videos quickly.

Furthermore, the existing ecosystem lacks a unified and integrated solution that combines every stage of the video summarization pipeline into one consolidated platform. Current tools generally handle tasks such as transcription, summarization, video processing, and user interaction in isolation, requiring users to navigate multiple interfaces or applications. This fragmentation not only increases complexity but also introduces inefficiencies and inconsistencies between different components. Without streamlined integration of audio extraction, transcription, NLP processing, summarization, and front end user control, the overall user experience suffers.

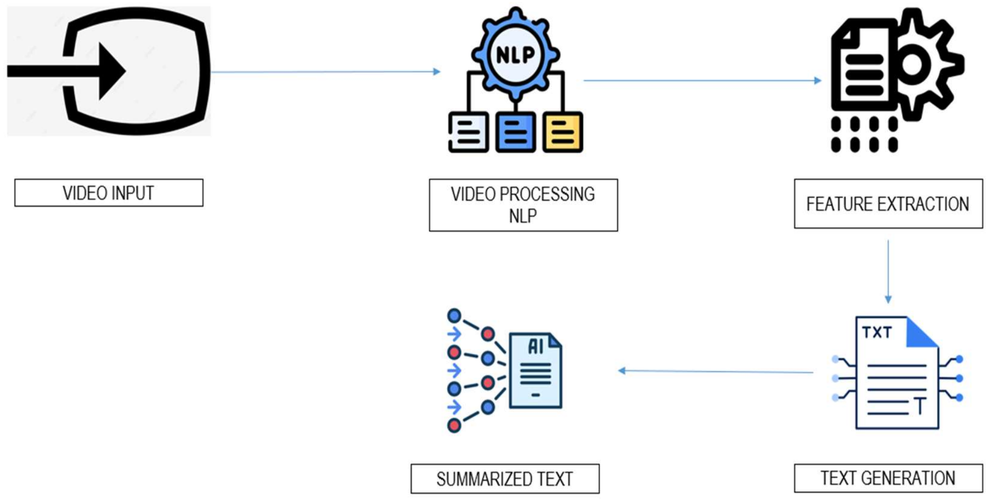


Figure 3.1 Existing System

CHAPTER – 4
PROBLEMS IDENTIFIED

- DIFFICULTY IN EXTRACTING MEANINGFUL INFORMATION FROM LONG VIDEOS:

  Users often find it challenging and time consuming to watch lengthy videos to identify key points. Many online videos contain lengthy introductions, irrelevant segments, or repetitive explanations, making it difficult to manually extract important information efficiently.

- DEPENDENCE ON ONLINE SERVICES FOR TRANSCRIPTION:

  Many existing systems rely on cloud based transcription APIs. This dependence causes issues such as internet unavailability, slow processing, privacy risks, and recurring subscription costs. In offline environments, these systems become unusable.

- INACCURATE OR NOISY AUTO GENERATED TRANSCRIPTS:

  Platforms like YouTube provide auto generated captions, but these often contain spelling errors, missing words, and timing mismatches. Such inaccuracies negatively affect the quality of summarization since the system relies on faulty input text.

- LACK OF INTEGRATED END TO END PROCESSING:

  Existing tools do not combine video extraction, audio processing, transcription, summarization, and output generation in one unified workflow. Users must manually switch between different applications, making the process inefficient and error prone.

- LIMITED CONTEXTUAL UNDERSTANDING IN TRADITIONAL NLP MODELS:

  Conventional summarization tools focus primarily on keyword extraction or sentence frequency. They lack true semantic understanding, resulting in summaries that are extractive, fragmented, and sometimes misleading.

- DIFFICULTY IN HANDLING SPOKEN LANGUAGE CHARACTERISTICS:

  Spoken content contains fillers, pauses, informal phrases, and non linear dialogue flow. Many summarization systems fail to correctly interpret such elements, causing loss of meaning, reduced coherence, and inaccurate output text.

- NO SUPPORT FOR OFFLINE, SECURE PROCESSING:

  Most existing tools process audio and text on external servers. Sensitive or private videos cannot be summarized securely. This creates a demand for an offline model that ensures privacy and data protection.

- LIMITED USER INTERACTION AND POOR INTERFACE USABILITY:

  Many tools lack a user friendly interface and instead require command line usage or multiple software installations. This reduces accessibility for non technical users who need a simple and intuitive summarization tool.

- PERFORMANCE ISSUES WITH LARGE OR HIGH QUALITY VIDEOS:

  Existing systems struggle with long duration or high resolution videos due to high processing requirements. Transcription delays, slow summarization speeds, and memory limitations often hinder smooth operation.

- INCONSISTENT SUMMARY QUALITY ACROSS DIFFERENT VIDEO TYPES:

  Videos vary widely in style, clarity, speaker accents, and content structure. Traditional summarization techniques fail to maintain consistent accuracy across lectures, tutorials, interviews, podcasts, and informal recordings, limiting their real world applicability.

CHAPTER – 5

PROPOSED SYSTEM

The proposed system presents a fully integrated, offline video summarization pipeline designed to convert long video content into concise, meaningful text summaries through accurate transcription and advanced natural language generation. The architecture brings together video processing, speech to text conversion, and AI driven summarization within a unified environment powered by Unity, Whisper, and Ollama. Unlike traditional systems that depend on cloud APIs or external services, the proposed solution operates entirely offline, ensuring privacy, reliability, and faster response times. The system is built to provide users with an easy and efficient method to extract essential information from lengthy videos without manually viewing the entire content.

The system consists of three major modules: the Video Processing and Audio Extraction Module, the Whisper Transcription Module, and the Ollama Summarization Module. All modules are seamlessly interconnected through Unity, which serves as the central user interface and coordination layer. The integration of these components ensures that raw video input can be transformed into high quality summaries with minimal user effort and without switching between multiple tools.

5.1 VIDEO PROCESSING AND AUDIO EXTRACTION MODULE

In this module, the user interacts with a Unity based interface to select the video that needs to be summarized. Unity handles the file input and communicates with Python scripts responsible for deeper processing. Python acts as the backend engine that retrieves the selected video and performs audio extraction. Through libraries such as MoviePy or FFmpeg, the system isolates the audio stream from the video file while maintaining clarity and synchronization.

This extracted audio serves as a clean input for the speech to text module. By directly operating on the raw audio, the system avoids limitations associated with external caption services or inaccurate transcripts. The module ensures that videos of different formats, resolutions, and lengths can be processed efficiently. This stage forms the foundation for accurate transcription, as high quality audio extraction directly influences the performance of the subsequent AI model.

5.2 WHISPER BASED TRANSCRIPTION MODULE

The extracted audio is passed to the Whisper transcription engine, a robust speech recognition model capable of handling diverse speech patterns, accents, background noise, and varying audio conditions. Whisper converts the audio into a detailed, accurate textual transcript. Unlike online based ASR services, Whisper processes everything locally, ensuring that sensitive or confidential content is never uploaded to external servers.

Whisper's transformer based architecture enables it to capture long range dependencies within speech, identify sentence boundaries, and preserve natural linguistic flow. This significantly reduces transcription errors and improves the quality of the raw text. The output transcript is structured, readable, and suitable for further summarization. The reliability of Whisper ensures that the final summary reflects the true meaning of the spoken content within the video.

5.3 OLLAMA ABSTRACTIVE SUMMARIZATION MODULE

After transcription, the text is processed by the summarization module powered by Ollama. In this module, a locally hosted large language model analyzes the transcript and generates a coherent, human like summary. Instead of selecting sentences from the transcript, the summarizer rephrases and reorganizes information to provide an abstractive summary that captures key points, essential concepts, and the overall intent of the video.

Ollama’s offline execution capabilities allow the summarization process to run without internet access, ensuring privacy and fast processing. The model understands context, identifies redundant segments, and condenses long passages into concise, meaningful output. This enables users to receive summaries that preserve both semantic richness and readability, making it easier to understand long or complex video content in a short span of time.

5.4 SYSTEM INTEGRATION AND FINAL OUTPUT

Unity acts as the central interface that integrates all modules and provides a smooth user experience. Once the summarization process is complete, Unity retrieves the output from Ollama and displays it within the application or exports it as a text file. This unified architecture eliminates the need for separate tools or manual intervention. The user receives a clean, well structured summary of the video with minimal effort.

Through the combination of local audio extraction, Whisper based transcription, Ollama driven summarization, and a Unity front end, the proposed system provides a highly efficient and reliable solution for video summarization. It supports offline operation, ensures data privacy, and produces high quality summaries that improve information accessibility and productivity for students, professionals, researchers, and general users.

5.5 BLOCK DIAGRAM OF PROPOSED SYSTEM

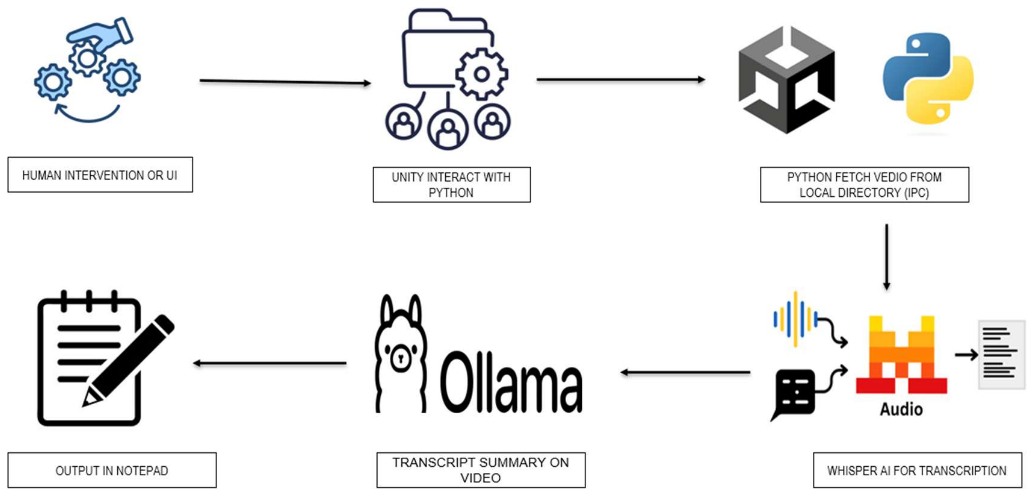

HUMAN INTERVENTION OR UI
UNITY INTERACT WITH PYTHON
PYTHON FETCH VEDIO FROM LOCAL DIRECTORY (IPC)
Audio
WHISPER AI FOR TRANSCRIPTION
Ollama
TRANSCRIPT SUMMARY ON VIDEO
OUTPUT IN NOTEPAD

Figure. 5.5.1 Proposed Block Diagram

5.6 ADVANTAGES:

- Accurate Extraction of Information from Long Videos:

  The system allows users to obtain precise and meaningful summaries from lengthy video content without manually watching the entire duration. This significantly enhances user efficiency by presenting only the essential information in a clear and concise format.

- Completely Offline Operation:

  All processes, including audio extraction, transcription, and summarization, run locally without the need for internet based services. This eliminates dependency on cloud APIs, reduces delays, and ensures consistent performance even in low connectivity environments.

- Enhanced Privacy and Data Securit:

  Since the system operates entirely offline, none of the audio or video content is uploaded to external servers. This ensures maximum confidentiality, making the system suitable for sensitive recordings such as academic lectures, internal meetings, and research discussions.

- High Quality Transcription Through Whisper:

  The use of Whisper provides robust transcription capabilities that remain accurate even in the presence of background noise, varied accents, or complex speech patterns. This improves the overall reliability of the generated summaries.

- Human Like Abstractive Summaries Using Ollama:

  Unlike traditional extractive methods, the system generates fluent, coherent, and context aware summaries. Ollama's language model restructures the transcript into meaningful text, offering high readability and better comprehension.

- Integrated and User Friendly Interface:

  Unity serves as a seamless front end platform, allowing users to upload videos, view processing progress, and retrieve summaries without navigating multiple applications. This unified workflow improves usability and provides a smooth user experience.

CHAPTER – 6

SYSTEM REQUIREMENTS

6.1 HARDWARE REQUIREMENTS

COMPONENT	SPECIFICATION
Processor	Intel i5 or above / AMD Ryzen
RAM	Minimum 4GB required
Storage	1GB free space for models and cache
GPU	NVIDIA GPU with CUDA support
Display	Standard monitor with 1080p or 720p display

6.2 SOFTWARE REQUIREMENTS

COMPONENT	SPECIFICATION
Operating System	Windows 10 or higher / macOS / Linux
Python Version	Python 3.9 or Python 3.10
Python Libraries	Whisper, MoviePy/FFmpeg, Ollama
Front End Engine	Unity 2021 or later
Summarization Engine	Ollama
Model Files	Whisper models, Ollama LLM model

CHAPTER – 7

SYSTEM IMPLEMENTATIONS

7.1 LIST OF MODULES

1. Video Data Collection & Preprocessing
2. Audio & Transcript Extraction
3. Text Preprocessing
4. Summarization Engine
5. Output Generation Interface

7.2 MODULES DESCRIPTION

The proposed system is composed of five core modules, each responsible for a critical stage in the overall video summarization workflow. These modules operate sequentially to ensure efficient processing, accurate transcription, and the generation of meaningful summaries. The following subsections describe the functionality, purpose, and internal operations of each module.

7.2.1 VIDEO DATA COLLECTION & PREPROCESSING

The system initiates its workflow by allowing the user to select a video file through an intuitive Unity based interface. This interface provides a simple and user friendly method for browsing local directories, ensuring that even non technical users can easily locate and choose the content they wish to summarize. Once a video is selected, Unity communicates the file path to the Python backend through an inter process communication mechanism. This connection ensures that the transition between the interface and backend processing remains seamless and fully automated.

After receiving the file path, the Python backend begins a series of validation checks to ensure the video is ready for processing. These checks include confirming that the file format is compatible with the system's processing tools and verifying that the video uses supported codecs for audio and visual streams. The system also inspects structural properties such as duration, frame rate, resolution, and general file stability. Identifying these characteristics early helps prevent errors that could occur later during audio extraction or transcription, especially in cases where corrupted, incomplete, or unsupported video files are encountered.

If the video is particularly large or encoded in a high resolution format, optional preprocessing steps may be applied to optimize performance. These optimizations include reducing redundant metadata, compressing large frame sequences when necessary, and ensuring the audio channel is clearly separated from the rest of the video.

7.2.2 AUDIO & TRANSCRIPT EXTRACTION

Once the video has been validated and prepared during preprocessing, the system proceeds to isolate the audio component, which forms the basis for transcription. The backend performs this operation using reliable multimedia processing tools such as FFmpeg or MoviePy. These tools extract the audio track directly from the video container while preserving its original quality, sampling rate, and synchronization. Ensuring that the extracted audio is free from distortions, clipping, or encoding artifacts is essential, as the precision of the speech to text conversion depends heavily on the clarity of the input signal. Any inconsistency or noise in the audio can interfere with recognition accuracy, so this stage emphasizes extracting a clean and well structured audio stream.

Once the audio track is ready, it is passed to Whisper, a state of the art offline speech recognition model based on advanced transformer architectures. Whisper is particularly well suited for this system because it can handle real world audio conditions, including background noise, overlapping speech, varied accents, rapid speech delivery, and shifts in tone or emphasis. Unlike traditional ASR systems that perform poorly in imperfect environments, Whisper is capable of understanding natural patterns of speech and converting them into coherent text. During transcription, the model not only identifies individual words but also interprets sentence boundaries, pauses, and contextual cues, producing a transcript that closely resembles human interpretation of spoken language.

The output produced at this stage is a raw transcript that captures the entire spoken narrative of the video in textual form. The transcript reflects not only the literal words spoken but also the flow of the conversation, the structure of explanations, and the transitions between different segments of the video. This textual representation becomes the foundation for all further processing steps in the summarization pipeline. Because the quality of the final summary depends directly on the accuracy and completeness of the transcript, Whisper's ability to maintain clarity, consistency, and contextual integrity plays a crucial role in achieving reliable results.

7.2.3 TEXT PREPROCESSING

Raw transcripts generated from spoken dialogue commonly include various imperfections such as pauses, repeated words, filler phrases, and inconsistent sentence structures. These issues are especially prominent in videos that are conversational, unscripted, or recorded in informal environments. Since human speech rarely follows strict grammatical rules, the raw transcript often appears fragmented, unorganized, and difficult to interpret in its initial form. Such irregularities can interfere with the summarization process, as the summarization engine relies heavily on clean and coherent text to understand context and extract meaning.

The text preprocessing stage addresses these challenges by transforming the raw transcript into a more organized and readable version. During this stage, the system identifies and removes unnecessary elements such as filler words, incomplete phrases, and background noise indicators that may have been mistakenly transcribed. Additional corrections include fixing inconsistent spacing, merging accidental line breaks, and eliminating residual characters introduced during transcription. These refinements help stabilize the structure of the text and eliminate distractions that could affect the model's interpretation.

Beyond basic cleaning, the transcript is further enhanced by reorganizing sentences into logical order and applying proper punctuation. This improves the grammatical flow and makes the text more closely resemble written language rather than raw speech patterns. Normalization techniques such as converting text to a uniform format, fixing capitalization, and correcting minor linguistic anomalies contribute to improving readability. The overall goal is to ensure that the transcript becomes clear, consistent, and semantically meaningful before it is passed to the summarization engine.

7.2.4 SUMMARIZATION ENGINE

Once the transcript has been refined and cleaned, it is passed to Ollama, where a locally hosted large language model carries out the task of abstractive summarization. At this stage, the model does not merely extract individual sentences or rank text segments based on frequency. Instead, it interprets the transcript holistically, analyzing the full context of the video content. The model examines the flow of ideas, identifies recurring themes, understands topic transitions, and recognizes which portions of the transcript carry the greatest informational value. This deeper level of interpretation allows it to grasp the underlying meaning rather than relying on superficial keyword matching.

During summarization, the model establishes connections between fragmented ideas scattered throughout the transcript. It reorganizes them into a cohesive structure and compresses the information into a shorter, more meaningful form. This involves rewriting sentences, merging similar concepts, and eliminating unnecessary repetition while preserving the intent of the speaker. The resulting summary reads naturally, capturing the essence of the video in language that resembles human writing rather than a stitched together set of extracted statements.

The strength of this approach lies in its ability to maintain clarity, continuity, and relevance throughout the condensed output. Even long, complex, or technical videos are distilled into clear and concise summaries that can be understood in a fraction of the original viewing time. Users no longer need to watch entire lectures, tutorials, or discussions to grasp the core ideas; the summarization engine delivers the main points in an organized and easily readable format.

By relying on contextual understanding and deep neural language modeling, the system ensures that the summary reflects both the factual content and the narrative flow of the original video. Intention, emphasis, and logical progression are preserved, allowing the summary to convey not just what was said, but why it was said. This advanced abstraction makes the summarization process significantly more effective than traditional extractive techniques and provides users with high quality summaries that remain faithful to the original meaning of the video

7.2.5 OUTPUT GENERATION INTERFACE

After the summarization is completed, Unity retrieves the generated text and displays it through a clear and organized interface. The summarized content is presented in a readable format so that users can easily understand the key points without navigating through complex menus or additional screens. The design focuses on simplicity and accessibility, ensuring that even non technical users can comfortably view and interpret the output.

Beyond on screen viewing, the system also provides the option to save the summary as a text file, typically opened through Notepad or any other lightweight text editor. This export feature allows users to store, edit, or share the summary for academic, professional, or personal use. It also supports smooth transfer of the summarized content across different devices or applications, making it convenient for long term reference.

The output module prioritizes clarity and usability by formatting the summary in a structured and visually comfortable layout. Proper spacing, alignment, and readability enhancements ensure that the final content is easy to review. This completes the overall workflow of the system by allowing users to obtain high quality summaries with minimal effort.

The interface makes the solution practical and beneficial for students reviewing lectures, educators preparing notes, professionals analyzing meeting recordings, or anyone frequently working with long form video content. By providing a simple yet effective way to access the final summary, the output module becomes a key element that enhances user experience and overall system efficiency.

CHAPTER – 8

SYSTEM TESTING

8.1 UNIT TESTING

Unit testing was conducted on each individual component of the video summarization system to verify that every module performed accurately in isolation before combining them into the full pipeline. The video loading function was tested to ensure Unity could correctly accept various file formats such as MP4, MKV, and AVI without producing errors. Python side video preprocessing functions were validated to confirm that metadata extraction, codec verification, and path handling worked reliably across multiple video samples. Audio extraction using FFmpeg was tested with videos of different durations, bitrates, and audio qualities to ensure that the resulting audio files remained clear and distortion free.

Whisper transcription was tested using audio samples that included background noise, varying speech speeds, multiple accents, and transitions between speakers. These unit tests confirmed that Whisper consistently generated accurate transcripts and handled edge cases such as silence intervals or clipped audio segments.

8.2 INTEGRATION TESTING

Integration testing evaluated how the system's modules interacted with one another when connected as an end to end workflow. The first integration check ensured smooth communication between Unity and the Python backend. Unity successfully passed the selected video path to Python, and Python returned status updates, allowing the front end interface to display progress indicators.

Next, audio extraction was integrated with Whisper transcription to confirm that the generated audio files were correctly processed into raw text. The quality and timing of the integration were tested to ensure that no data was lost or interrupted during handover. The integration of text preprocessing with the summarization module was also evaluated. Cleaned transcripts were successfully fed into the summarization engine, and the summarizer generated outputs that retained structural coherence.

This complete cycle was tested using different categories of videos such as lectures, tutorials, discussions, and meetings. Special attention was given to time synchronization and the ability of the system to trigger summarization immediately after transcription. Integration testing also ensured that the system could handle rapid sequential processing, meaning multiple videos could be summarized in succession without requiring a restart. The absence of conflicting processes, stalled modules, or broken communication channels confirmed the successful integration of all components.

8.3 SYSTEM TESTING

System testing examined the performance of the entire application under realistic usage conditions to ensure that it met all specified functional requirements. Multiple test videos—ranging from short clips to long format content—were used to evaluate the completeness and reliability of the summarization workflow.
Tests confirmed that the system could consistently load videos, extract audio, generate transcripts, and produce accurate summaries without crashes or data loss. Whisper's transcription accuracy was validated under conditions that included low quality audio, background noise, and multi speaker input. The summarizer was tested to ensure it retained essential points, followed logical order, and produced readable summaries across diverse content types.

Long duration system tests were carried out to check for memory leaks, processing bottlenecks, or performance drops during extended usage. The system was also tested while other applications ran in the background to evaluate stability under multitasking scenarios. Repeated start–stop cycles validated that the system could initialize and terminate smoothly, without leaving residual processes or corrupted temporary files. Overall, system testing demonstrated that the application performed reliably as a complete offline summarization tool.

8.4 PERFORMANCE TESTING

Performance testing focused on evaluating the speed, responsiveness, and efficiency of the system during continuous operation. Audio extraction speed was measured across videos of varying lengths, confirming that the system maintained stable processing times without significant delays. Whisper transcription was assessed in terms of accuracy and latency, with most transcripts generated within reasonable time based on video duration and hardware specifications.

The summarization engine was evaluated for its ability to handle large transcripts without slowdowns or interruptions. Time taken to generate summaries was monitored to ensure consistent performance across different workloads. CPU and RAM usage were measured during simultaneous execution of transcription and summarization to guarantee that the system operated efficiently without exhausting system resources.
Additional performance tests involved running the system with larger video files, high resolution content, or videos with heavy background noise. These variations helped confirm that the system remained stable and responsive even under challenging conditions. Stress testing was also performed by summarizing multiple videos consecutively to assess how well the system responded under high load.

CHAPTER – 9

RESULT AND DISCUSSION

The proposed Offline Video Summarization System was successfully developed and evaluated to determine its effectiveness in generating concise and meaningful summaries from long form video content. The system demonstrated strong performance across all stages of the pipeline, including video selection through Unity, audio extraction, Whisper based transcription, and abstractive summarization using Ollama. By integrating these technologies, the system provided a seamless end to end summarization experience without requiring internet connectivity, ensuring both privacy and reliability during operation.

During testing, the audio extraction component consistently produced clear and distortion free audio files from videos of various formats and resolutions. Whisper achieved highly accurate speech to text conversion across a wide range of test videos, including academic lectures, tutorials, interviews, and discussion based content. This accuracy remained stable even when videos contained slight background noise or speakers with different accents. The transcription quality was notably strong, preserving sentence boundaries and maintaining contextual flow, which significantly contributed to generating meaningful summaries later in the pipeline.

The summarization module powered by Ollama displayed excellent capability in condensing long transcripts into coherent and human like summaries. Rather than extracting isolated sentences, the model consistently delivered structured explanations that captured the main ideas, important points, and transitions present in the original video. Users found that the abstraction effectively reduced lengthy content into concise paragraphs that were easy to understand, making it particularly valuable for time sensitive review tasks. Additionally, the summarization process remained stable for both short and long transcripts, confirming the model’s robustness in handling diverse workloads.

One of the most significant outcomes observed was the system’s reliable performance under varying input conditions. Videos with strong audio clarity delivered optimal results; however, even videos recorded in moderately noisy environments produced accurate transcripts and high quality summaries. Challenges arose only in scenarios where audio was extremely unclear or when the speaker’s voice was too distant from the microphone. In such extreme cases, slight inaccuracies were seen at the transcription level, which led to mildly reduced summary precision. Nonetheless, normal and moderately challenging conditions did not significantly affect the output quality.

User evaluations indicated that the system was easy to operate due to its clean Unity based interface. Participants appreciated the straightforward workflow: selecting a video, waiting for processing, and receiving a formatted summary without navigating complex menus or performing manual steps. The ability to export summaries into text files was found to be especially useful for academic note taking, professional documentation, and quick reference. Users also highlighted the value of offline functionality, noting that the system ensured complete privacy, particularly when processing sensitive or confidential video content.

Performance testing showed that the system remained stable during long duration use. Whisper and Ollama both operated smoothly without noticeable memory leaks or slowdowns, even when summarizing multiple videos consecutively. Processing time increased proportionally with video length, but remained efficient enough for practical use.Overall, the results confirm that the offline summarization system is a practical, efficient, and user friendly solution for extracting essential information from video content. The integration of Whisper and Ollama provides high transcription accuracy and high quality abstractive summaries, while Unity ensures ease of use and a clean interaction environment. Although extreme audio conditions or heavily accented speech may occasionally affect transcription quality, the system performs exceptionally well in typical real world scenarios.

CHAPTER – 10

CONCLUSION AND FUTURE WORK

10.1 CONCLUSION

The Offline Video Summarization System successfully demonstrates the potential of combining modern AI models with a unified interaction framework to provide an efficient and reliable method for understanding long form video content. By integrating Unity for user interaction, Python for backend processing, Whisper for offline speech to text conversion, and Ollama for abstractive summarization, the project delivers a smooth and automated workflow that transforms raw video into concise, meaningful textual summaries. The system eliminates the need for manual note taking or watching lengthy videos in full, making it a highly practical tool for students, educators, professionals, and researchers who require quick access to essential information.

Throughout testing, the system consistently produced accurate transcripts and coherent summaries across various types of videos, including academic lectures, tutorials, interviews, and discussions. Whisper demonstrated strong transcription performance even with moderate background noise or varied speaker accents, while the summarization engine generated clear and contextually rich summaries that effectively captured the key ideas of each video. The Unity interface further enhanced usability by providing a simple and accessible environment for selecting videos and viewing results. These outcomes highlight the system’s practicality, reliability, and ease of use in real world scenarios.

Although certain limitations were observed—such as reduced transcription accuracy in extremely noisy audio and slightly increased processing time for very long videos—the overall performance confirms that offline summarization is both feasible and highly effective. The ability to operate without internet dependency ensures complete privacy and makes the system suitable for sensitive or confidential video content.

Overall, the project demonstrates a modern and efficient approach to video understanding, showcasing how AI based transcription and summarization can significantly enhance information accessibility. The success of this system establishes a strong foundation for future advancements, including multilingual support, improved summarization control, enhanced audio cleaning, topic based segmentation, and real time summarization capabilities.

10.2 FUTURE ENHANCEMENTS

Although the current system provides a reliable and efficient method for summarizing video content offline, several future enhancements can further expand its capability, accuracy, and usability. The following improvements have the potential to elevate the system into a more advanced and versatile summarization platform:

1. **MULTILINGUAL TRANSCRIPTION AND SUMMARIZATION:**

   Future versions can incorporate multilingual Whisper models and corresponding summarization models in Ollama to support videos recorded in regional and international languages. This enhancement would allow the system to process a broader range of educational, professional, and global media content, making it more inclusive and useful for diverse users.

2. **CUSTOMIZABLE SUMMARY LENGTH AND STYLE:**

   Adding options for users to choose between short, medium, or detailed summaries can significantly improve flexibility. Advanced models may also allow summaries in different styles such as bullet points, analytical summaries, or descriptive narratives, enabling the system to cater to academic, professional, and research needs.

3. **TOPIC BASED SEGMENTATION AND CHAPTER WISE SUMMARIES:**

Integrating intelligent segmentation techniques can divide long videos into logical sections or chapters. Each section can then be summarized individually, providing users with structured insights rather than a single long summary.

4. **ENHANCED AUDIO CLEANING AND NOISE REDUCTION:**

Incorporating advanced audio enhancement techniques such as noise filtering, echo cancellation, and speech isolation can improve transcription accuracy in videos recorded in noisy or low quality environments. This would allow the system to perform more consistently even under challenging audio conditions.

5. **REAL TIME OR NEAR REAL TIME SUMMARIZATION:**

Future upgrades may include streaming based processing where transcription and summarization occur while the video is playing. This would reduce waiting time and make the system suitable for live lectures, online classes, webinars, and real time monitoring applications.

6. **INTEGRATION WITH CLOUD AND HYBRID PROCESSING**

Although the system is designed to operate fully offline, optional cloud integration could be provided for users who prefer faster processing or require access to larger, more powerful models. A hybrid mode could intelligently switch between offline and online processing depending on user preferences and available resources.

7. **ADVANCED USER INTERFACE WITH SUMMARY COMPARISON**

Enhancing the Unity interface to include features such as side by side summary comparison, keyword extraction visualization, and transcript highlighting would give users deeper insight into the summarized content. These tools can support students and researchers in analyzing differences across multiple videos or multiple summarization outputs.

APPENDIX – A

SOURCE CODE

main.py

```
import requests

import whisper

import os

from moviepy.video.io.VideoFileClip import VideoFileClip

import time

while(True):

  try:

    var=""

    file=None


    def check():

      while(True):


        if(os.path.exists("D:/AiPath/samples.txt")):

          print("found something")

          global file

          file=open("D:/AiPath/samples.txt","r")

          global var

          var=file.read()

          file.close()
```

```
            files= open("D:/auth/fileFound.txt","w")
            files.close()


            break


        else:
            time.sleep(0.5)
            print("nothing")
check()
try:
    name=var.strip()
    vid=VideoFileClip(f"D:/vid/{name}")
    vid.audio.write_audiofile("aud.wav")
    obj=whisper.load_model("small")
    text=obj.transcribe("aud.wav")
    prompt=text["text"]
    file.close()
    files= open("D:/auth/transcription.txt","w")
    files.close()
    obj={
        "prompt":prompt+"remove ** in response"+"summarize this text",
        "model":"llama3.2",
        "stream":False
    }
```

```
            req=requests.post("http://localhost:11434/api/generate",json=obj)
            ans=req.json().get("response")
            #answer=ans.split("</think>")[1]
        except(FileExistsError):
            var.strip()
            print(FileExistsError)
        os.remove("D:/AiPath/samples.txt")
        notes=open("D:/notes/notes.txt","w")
        files= open("D:/auth/saved.txt","w")
        files.close()
        notes.write(ans)
    except(Exception):
        open("D:/auth/error.txt","w")
        pass
```

connect.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;
using System;
using TMPro;
public class FileREad : MonoBehaviour
{
    string content = "hello";
    string path = "D:/AiPath/samples.txt";
    public Text txts;
```

```
List<string> files = new List<string>();
public Button btn;
public Text t;
int timers = 10;
int num = 0;
public Animator animator;
public GameObject stool;
int auth1 = 0;
int auth2 = 0;
int auth3 = 0;
public TextMeshProUGUI textMeshPro;
void Start()
{
   btn.onClick.AddListener(textFile);
   t.text = "";
   stool.SetActive(!true);

}

private void Update()
{
   if(num==1)
   {
      animator.SetBool("work", true);
   }
   if (timers == 0)
   {
      animator.SetBool("work", !true);
      num = 0;


   }
```

```
    if(File.Exists("D:/auth/fileFound.txt")&&auth1==0)
    {
        textMeshPro.text = "WE FOUND YOUR VIDEO FILE...";
        File.Delete("D:/auth/fileFound.txt");

    }
    if (File.Exists("D:/auth/transcription.txt") && auth2 == 0)
    {
        textMeshPro.text = "TRANSCRIPTION DONE";
        File.Delete("D:/auth/transcription.txt");



    }
    if (File.Exists("D:/auth/saved.txt") && auth3 == 0)
    {
        textMeshPro.text = "WE FOUND YOUR VIDEO FILE...";
        File.Delete("D:/auth/saved.txt");



    }
    if (File.Exists("D:/auth/error.txt") && auth3 == 0)
    {
        textMeshPro.text = "SORRY THERE IS AN ERROR WE WILL FIX IT";
        File.Delete("D:/auth/error.txt");



    }
}

void textFile()
{
    if (txts.text != "")
    {
        stool.SetActive(true);
        files.Add(txts.text);
        Debug.Log("ok");
        File.WriteAllLines(path, files);
        num = 1;
        timers = 100;
        StartCoroutine("timer");
    }
    else
    {

        Debug.Log("!ok");
    }
}
IEnumerator timer()
{
    while (true)
    {

        if(num==1&&timers!=0)
        {

            yield return new WaitForSeconds(1);
            timers  = 1;
            t.text=timers.ToString()+"s";

        }

        else
```

```
            {

                //yield return new WaitForSeconds(1);
                break;
            }
        }
    }
}
```

Library.cs

```
using System;
using System.Collections;
using System.Runtime.CompilerServices;
using System.Threading;
using UnityEngine.Bindings;
using UnityEngine.Internal;
using UnityEngine.Scripting;

namespace UnityEngine
{
    [RequiredByNativeCode]
    [NativeHeader("Runtime/Scripting/DelayedCallUtility.h")]
    [NativeHeader("Runtime/Mono/MonoBehaviour.h")]
    [ExtensionOfNativeClass]
    public class MonoBehaviour : Behaviour
    {
        private CancellationTokenSource m_CancellationTokenSource;

        public CancellationToken destroyCancellationToken
        {
            get
    {
        if (m_CancellationTokenSource == null)
        {
            m_CancellationTokenSource = new CancellationTokenSource();
            OnCancellationTokenCreated();
        }
        return m_CancellationTokenSource.Token;
    }
}

public extern bool useGUILayout
{
    [MethodImpl(MethodImplOptions.InternalCall)]
    get;
    [MethodImpl(MethodImplOptions.InternalCall)]
    set;
}

public extern bool runInEditMode
{
    [MethodImpl(MethodImplOptions.InternalCall)]
    get;
    [MethodImpl(MethodImplOptions.InternalCall)]
    set;
}

internal extern bool allowPrefabModeInPlayMode
{
    [MethodImpl(MethodImplOptions.InternalCall)]
    get;
}
```

```
public MonoBehaviour()
{
   ConstructorCheck(this);
}

[RequiredByNativeCode]
private void RaiseCancellation()
{
   m_CancellationTokenSource?.Cancel();
}

public bool IsInvoking()
{
   return Internal_IsInvokingAll(this);
}

public void CancelInvoke()
{
   Internal_CancelInvokeAll(this);
}

public void Invoke(string methodName, float time)
{
   InvokeDelayed(this, methodName, time, 0f);
}

public void InvokeRepeating(string methodName, float time, float repeatRate)
{
   if (repeatRate <= 1E 05f && repeatRate != 0f)
   {
```

```
            throw new UnityException("Invoke repeat rate has to be larger than
0.00001F)");
        }
        InvokeDelayed(this, methodName, time, repeatRate);
    }

    public void CancelInvoke(string methodName)
    {
        CancelInvoke(this, methodName);
    }

    public bool IsInvoking(string methodName)
    {
        return IsInvoking(this, methodName);
    }

    public Coroutine StartCoroutine(string methodName)
    {
        object value = null;
        return StartCoroutine(methodName, value);
    }

    public Coroutine StartCoroutine(string methodName, object value)
    {
        if (string.IsNullOrEmpty(methodName))
        {
            throw new NullReferenceException("methodName is null or empty");
        }
        if (!IsObjectMonoBehaviour(this))
        {
            throw new ArgumentException("Coroutines can only be stopped on a
MonoBehaviour");
         }
         return StartCoroutineManaged(methodName, value);
      }

      public Coroutine StartCoroutine(IEnumerator routine)
      {
         if (routine == null)
         {
            throw new NullReferenceException("routine is null");
         }
         if (!IsObjectMonoBehaviour(this))
         {
            throw new ArgumentException("Coroutines can only be stopped on a
MonoBehaviour");
         }
         return StartCoroutineManaged2(routine);
      }

      [Obsolete("StartCoroutine_Auto has been deprecated. Use StartCoroutine instead
(UnityUpgradable)  > StartCoroutine([mscorlib] System.Collections.IEnumerator)",
false)]
      public Coroutine StartCoroutine_Auto(IEnumerator routine)
      {
         return StartCoroutine(routine);
      }

      public void StopCoroutine(IEnumerator routine)
      {
         if (routine == null)
```

```
        {
            throw new NullReferenceException("routine is null");
        }
        if (!IsObjectMonoBehaviour(this))
        {
            throw new ArgumentException("Coroutines can only be stopped on a
MonoBehaviour");
        }
        StopCoroutineFromEnumeratorManaged(routine);
    }

    public void StopCoroutine(Coroutine routine)
    {
        if (routine == null)
        {
            throw new NullReferenceException("routine is null");
        }
        if (!IsObjectMonoBehaviour(this))
        {
            throw new ArgumentException("Coroutines can only be stopped on a
MonoBehaviour");
        }
        StopCoroutineManaged(routine);
    }

    [MethodImpl(MethodImplOptions.InternalCall)]
    public extern void StopCoroutine(string methodName);

    [MethodImpl(MethodImplOptions.InternalCall)]
    public extern void StopAllCoroutines();
    public static void print(object message)
    {
      Debug.Log(message);
    }

    [MethodImpl(MethodImplOptions.InternalCall)]
    [NativeMethod(IsThreadSafe = true)]
    private static extern void ConstructorCheck([Writable] Object self);

    [MethodImpl(MethodImplOptions.InternalCall)]
    [FreeFunction("CancelInvoke")]
    private static extern void
Internal_CancelInvokeAll([NotNull("NullExceptionObject")] MonoBehaviour self);

    [MethodImpl(MethodImplOptions.InternalCall)]
    [FreeFunction("IsInvoking")]
    private static extern bool
Internal_IsInvokingAll([NotNull("NullExceptionObject")] MonoBehaviour self);

    [MethodImpl(MethodImplOptions.InternalCall)]
    [FreeFunction]
    private static extern void InvokeDelayed([NotNull("NullExceptionObject")]
MonoBehaviour self, string methodName, float time, float repeatRate);

    [MethodImpl(MethodImplOptions.InternalCall)]
    [FreeFunction]
    private static extern void CancelInvoke([NotNull("NullExceptionObject")]
MonoBehaviour self, string methodName);

    [MethodImpl(MethodImplOptions.InternalCall)]
    [FreeFunction]
```

```
        private static extern bool IsInvoking([NotNull("NullExceptionObject")]
MonoBehaviour self, string methodName);

        [MethodImpl(MethodImplOptions.InternalCall)]
        [FreeFunction]
        private static extern bool
IsObjectMonoBehaviour([NotNull("NullExceptionObject")] Object obj);

        [MethodImpl(MethodImplOptions.InternalCall)]
        private extern Coroutine StartCoroutineManaged(string methodName, object
value);

        [MethodImpl(MethodImplOptions.InternalCall)]
        private extern Coroutine StartCoroutineManaged2(IEnumerator enumerator);

        [MethodImpl(MethodImplOptions.InternalCall)]
        private extern void StopCoroutineManaged(Coroutine routine);

        [MethodImpl(MethodImplOptions.InternalCall)]
        private extern void StopCoroutineFromEnumeratorManaged(IEnumerator
routine);

        [MethodImpl(MethodImplOptions.InternalCall)]
        internal extern string GetScriptClassName();

        [MethodImpl(MethodImplOptions.InternalCall)]
        private extern void OnCancellationTokenCreated();
    }
}
```

APPENDIX – B

SCREENSHOTS

Sample Output

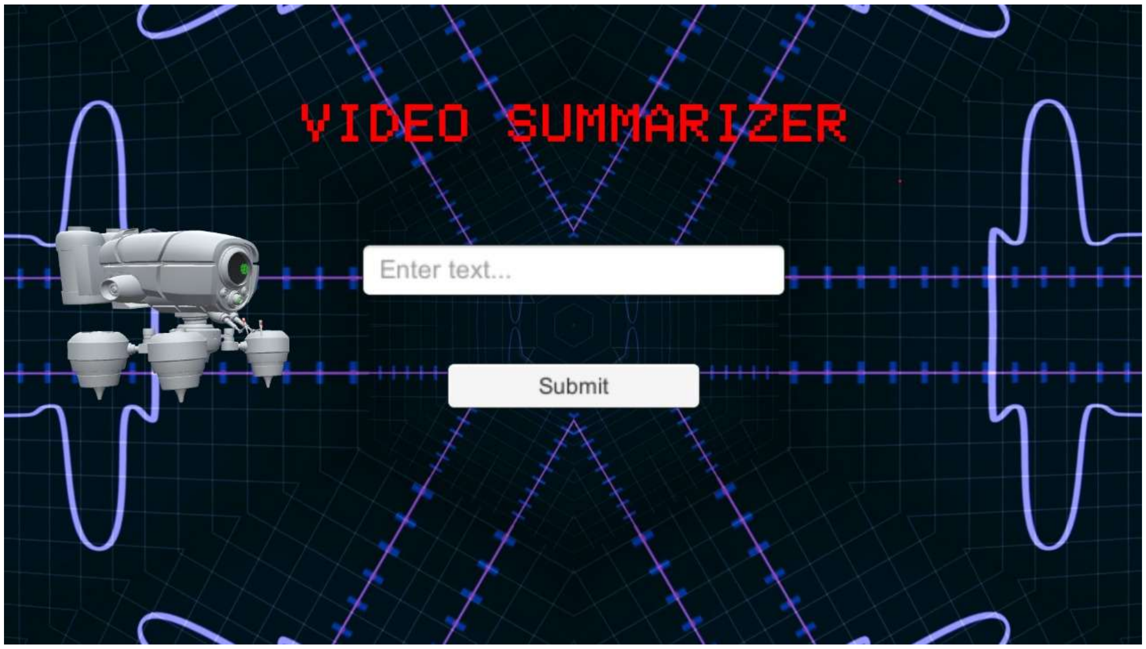
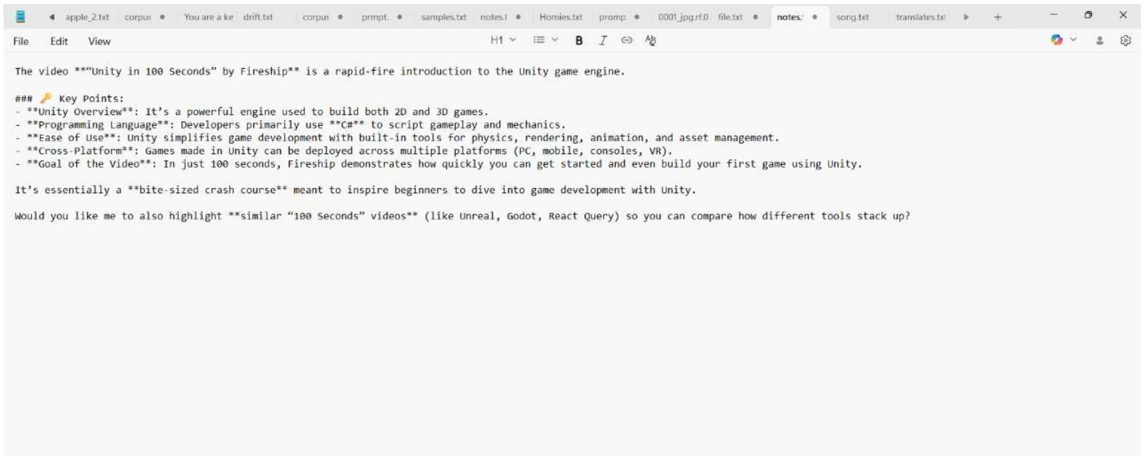


Figure. B.1. Home Page



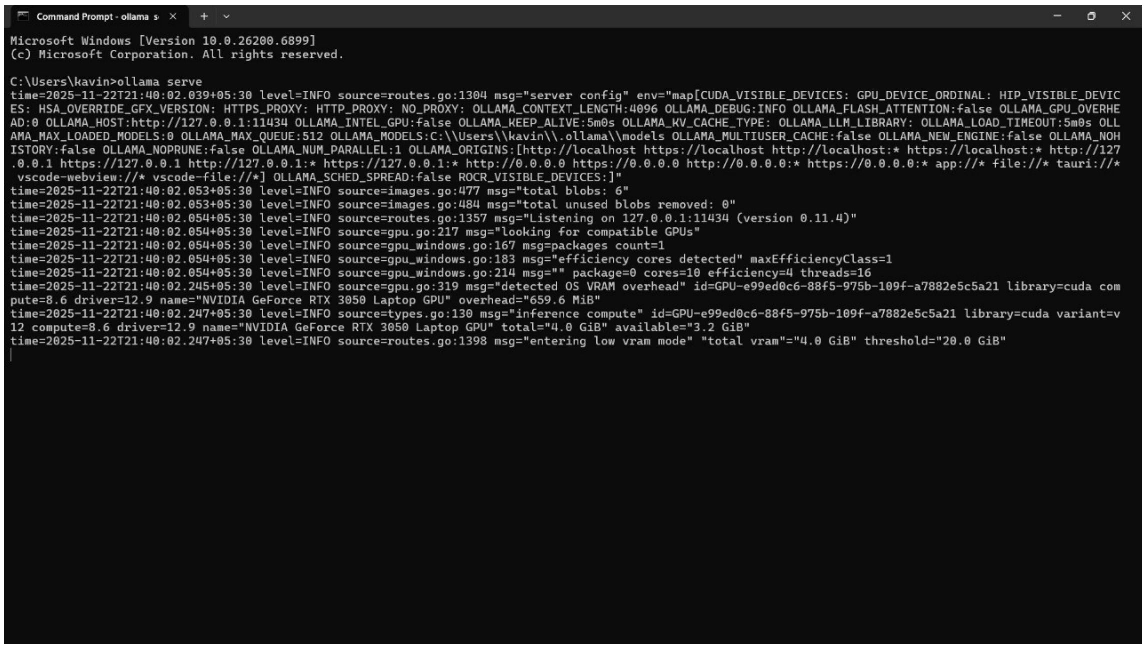
```
The video **“Unity in 100 Seconds” by Fireship** is a rapid-fire introduction to the Unity game engine.

### 🔑 Key Points:
- **Unity Overview**: It’s a powerful engine used to build both 2D and 3D games.
- **Programming Language**: Developers primarily use **C#** to script gameplay and mechanics.
- **Ease of Use**: Unity simplifies game development with built-in tools for physics, rendering, animation, and asset management.
- **Cross-Platform**: Games made in Unity can be deployed across multiple platforms (PC, mobile, consoles, VR).
- **Goal of the Video**: In just 100 seconds, Fireship demonstrates how quickly you can get started and even build your first game using Unity.

It’s essentially a **bite-sized crash course** meant to inspire beginners to dive into game development with Unity.

Would you like me to also highlight **similar “100 Seconds” videos** (like Unreal, Godot, React Query) so you can compare how different tools stack up?
```

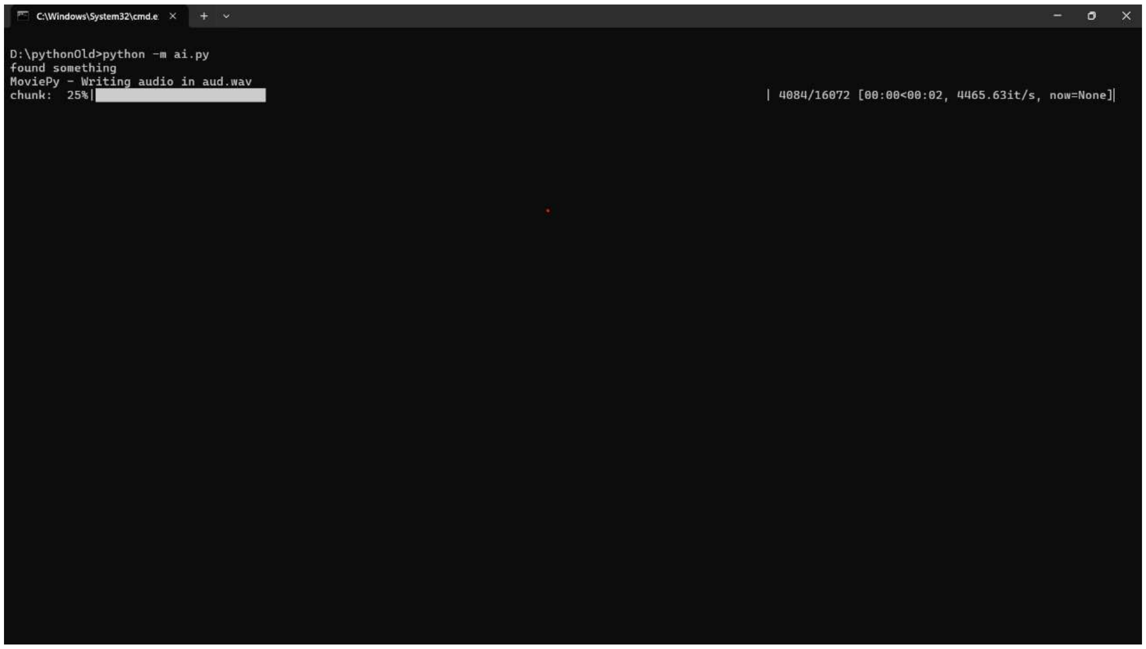
Figure. B.2. Summary File



```
Microsoft Windows [Version 10.0.26200.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kavin>ollama serve
time=2025-11-22T21:40:02.039+05:30 level=INFO source=routes.go:1304 msg="server config" env="map[CUDA_VISIBLE_DEVICES: GPU_DEVICE_ORDINAL: HIP_VISIBLE_DEVIC
ES: HSA_OVERRIDE_GFX_VERSION: HTTPS_PROXY: HTTP_PROXY: NO_PROXY: OLLAMA_CONTEXT_LENGTH:4096 OLLAMA_DEBUG:INFO OLLAMA_FLASH_ATTENTION:false OLLAMA_GPU_OVERHE
AD:0 OLLAMA_HOST:http://127.0.0.1:11434 OLLAMA_INTEL_GPU:false OLLAMA_KEEP_ALIVE:5m0s OLLAMA_KV_CACHE_TYPE: OLLAMA_LLM_LIBRARY: OLLAMA_LOAD_TIMEOUT:5m0s OLL
AMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELS:C:\\Users\\kavin\\.ollama\\models OLLAMA_MULTIUSER_CACHE:false OLLAMA_NEW_ENGINE:false OLLAMA_NOH
ISTORY:false OLLAMA_NOPRUNE:false OLLAMA_NUM_PARALLEL:1 OLLAMA_ORIGINS:[http://localhost https://localhost http://localhost:* https://localhost:* http://127
.0.0.1 https://127.0.0.1 http://127.0.0.1:* https://127.0.0.1:* http://0.0.0.0 https://0.0.0.0 http://0.0.0.0:* https://0.0.0.0:* app://* file://* tauri://*
 vscode-webview://* vscode-file://*] OLLAMA_SCHED_SPREAD:false ROCR_VISIBLE_DEVICES:]"
time=2025-11-22T21:40:02.053+05:30 level=INFO source=images.go:477 msg="total blobs: 6"
time=2025-11-22T21:40:02.053+05:30 level=INFO source=images.go:484 msg="total unused blobs removed: 0"
time=2025-11-22T21:40:02.054+05:30 level=INFO source=routes.go:1357 msg="Listening on 127.0.0.1:11434 (version 0.11.4)"
time=2025-11-22T21:40:02.054+05:30 level=INFO source=gpu.go:217 msg="looking for compatible GPUs"
time=2025-11-22T21:40:02.054+05:30 level=INFO source=gpu_windows.go:167 msg=packages count=1
time=2025-11-22T21:40:02.054+05:30 level=INFO source=gpu_windows.go:183 msg="efficiency cores detected" maxEfficiencyClass=1
time=2025-11-22T21:40:02.054+05:30 level=INFO source=gpu_windows.go:214 msg="" package=0 cores=10 efficiency=4 threads=16
time=2025-11-22T21:40:02.245+05:30 level=INFO source=gpu.go:319 msg="detected OS VRAM overhead" id=GPU-e99ed0c6-88f5-975b-109f-a7882e5c5a21 library=cuda com
pute=8.6 driver=12.9 name="NVIDIA GeForce RTX 3050 Laptop GPU" overhead="659.6 MiB"
time=2025-11-22T21:40:02.247+05:30 level=INFO source=types.go:130 msg="inference compute" id=GPU-e99ed0c6-88f5-975b-109f-a7882e5c5a21 library=cuda variant=v
12 compute=8.6 driver=12.9 name="NVIDIA GeForce RTX 3050 Laptop GPU" total="4.0 GiB" available="3.2 GiB"
time=2025-11-22T21:40:02.247+05:30 level=INFO source=routes.go:1398 msg="entering low vram mode" "total vram"="4.0 GiB" threshold="20.0 GiB"
```

Figure. B.3. Server Initialization



```
D:\pythonOld>python -m ai.py
found something
MoviePy - Writing audio in aud.wav
chunk:  25%|████████████████████                                                                  | 4084/16072 [00:00<00:02, 4465.63it/s, now=None]
```

Figure. B.4. Interpretation Between Unity and LLM

REFERENCES

1. Aniqa Dilawari, Muhammad Usman Ghani Khan, "ASoVS: Abstractive Summarization of Video Sequences," IEEE Access*,* vol. 7, pp. 29253–29263, 2019.
2. Cycle SUM: Cycle consistent Adversarial LSTM Networks for Unsupervised Video Summarization Li Yuan, Francis EH Tay, Ping Li, Li Zhou, Jiashi Feng, AAAI 2019
3. Deep Reinforcement Learning for Unsupervised Video Summarization Kaigang Zhou, Yu Qiao, Tao Xiang, AAAI 2018
4. Extractive Automatic Text Summarization using SpaCy Swaranjali Jugran, Ashish Kumar, Bhupendra S. Tyagi, Vivek Anand, ICAITE 2021
5. Fully Convolutional Sequence Networks for Video Summarization Mrigank Rochan, Linwei Ye, Yang Wang, ECCV 2018
6. Hierarchical Recurrent Neural Network for Video Summarization Bin Zhao, Xuelong Li, Xiaoqiang Lu, ACM Multimedia, 2017
7. Method of Text Summarization using LSA and BERT Hritvik Gupta and Mayank Patel, IEEE ICAIS 2021
8. Unsupervised Video Summarization with Adversarial LSTM Networks (SUM GAN) Behrooz Mahasseni, Michael Lam, Sinisa Todorovic, CVPR 2017
9. YouTube Video Summarizer Using NLP: A Review Yogendra Singh, Rishu Kumar, Soumya Kabdal, Prashant Upadhyay, International Journal of Performability Engineering, 2023.
10. Yogendra Singh, Rishu Kumar, Soumya Kabdal, Prashant Upadhyay, "YouTube Video Summarizer Using NLP: A Review," International Journal of Performability Engineering*,* vol. 19, no. 12, pp. 817–823, 2023.