AUDIO SUMMARIZATION IN REAL TIME FOR PODCAST, SPEECHES AND AUDIO BOOKS

A PROJECT REPORT

Submitted by

LOGESH KUMAR D [211419104148]
NARENDHIRAN S [211419104176]
HEMANTH S [211419104101]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonmous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"AUDIO SUMMARIZATION IN REAL TIME FOR PODCAST, SPEECHES AND AUDIO BOOKS"** is the bonafide work of **" LOGESH KUMAR D (211419104148), NARENDHIRAN S (211419104176), HEMANTH S (211419104101)"** who carried out the project work under my supervision.

SIGNATURE

Dr.L. JABASHEELA, M.E., Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mr.A.N.SASIKUMAR
SUPERVISOR
ASSISTANT PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on **11-04-2023**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **LOGESH KUMAR D (211419104148), NARENDHIRAN S (211419104176), HEMANTH S (211419104101)** hereby declare that this project report titled **"AUDIO SUMMARIZATION IN REAL TIME FOR PODCAST, SPEECHES AND AUDIO BOOKS"** under the guidance of **Mr.A.N.SASIKUMAR,M.E.,** is the orginial work done by us and we have not plagiarized or submitted to any other degree in any university by us.

LOGESH KUMAR D

NARENDHIRAN S

HEMANTH S

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L. JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank my **Pr Mr.A.N.SASIKUMAR, M.E.,** and coordinator **Dr.G.SENTHIL KUMAR, M.C.A.,M.Phil.,M.B.A., M.E., Ph.D.,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

LOGESH KUMAR D

NARENDHIRAN S

HEMANTH S

ABSTRACT

Due to the coronavirus disease (COVID-19) pandemic, most of the public work is carried out online. Universities all around the globe moved to online education, job interviews are mainly conducted online, many first-level health consultations are happening online, and companies hold periodic meetings entirely online. Google Meet, Microsoft Team, and other online meeting software applications are widely accessible on the market. In this work, we are addressing a topic that has a lot of practical applications. In this paper, we present a method that takes a recorded video as an input and generates a written and/or audio summary of the same as an output. The suggested method can also be used to generate lecture notes from lecture videos, meeting minutes, subtitles, or storyline production from entertainment videos, among several other things. The suggested system takes the video's audio track, which is then transformed into text. In addition, we created the text summary utilizing text summarising algorithms. The system's users have the option of using the text summary or creating an audio output that matches the text summary. The proposed method is implemented in Python, and the proposed scheme is evaluated using short videos acquired from YouTube. Since there is no benchmark measure for evaluating the efficiency and there is no specific dataset available for the relevant study, the proposed method is manually validated on the downloaded video set.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

ABB		EXPANSION
OCR	-	Optical Character Recognition
NLP	-	Natural Language Processing
ASR	-	Automatic Speech Recognition
TTS	-	Text-to-Speech
ML	-	Machine Learning
GUI	-	Graphical User Interface
OCR	-	Optical Character Recognition
NLP	-	Natural Language Processing

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Audio summarization is the process of extracting the essential information from an audio recording and presenting it in a concise format. With the growing popularity of podcasts, speeches, and audio books, audio summarization has become increasingly important. Real-time audio summarization is a subset of audio summarization that allows for the quick extraction and presentation of information as it is being spoken. In this article, we will explore audio summarization in real-time for podcast, speeches, and audio books, its applications, tools, advantages, challenges, and future possibilities.

Interest in immersive communication technologies has been growing over the last two decades due to the emergence of today's multimedia applications. Gaming, virtual and augmented reality (VR and AR), teleconferencing, and entertainment applications have taken a big step forward due to the spread of mobile multimedia and ubiquitous computing. Spatial audio research is at the center of the new developments in 3D immersive user experiences, providing the core technologies for spatial sound capture, processing, and reproduction.

Spatial audio is an interdisciplinary field of research that brings together experts from audio engineering, acoustics, computer science, applied psychoacoustics, and other domains. The aim of spatial audio is to recreate an acoustic environment or synthesize a new one by using a proper combination of sound recording, processing, and reproduction techniques. Within such an objective, it is not only important to preserve the fidelity of the audio content but also the spatial attributes of the sound scene resulting from the actual locations of the sound

sources and the properties of the acoustic environment.

In general, spatial audio methods and techniques may be broadly described. complete spatial audio pipeline generally comprises a capture stage, a processing stage in which the spatial information in the captured sound scene is modified or in which given information is extracted from the sound scene that cannot be measured directly, and finally a reproduction stage in which the (potentially manipulated) sound scene is ruralized.

Summarization is the technique of condensing data to a form that is quick to peruse, easy to understand and, the technique of conveying the crux of the content whilst maintaining the grammar and semantics of the language. Optical character recognition refers to the process of identifying text from imagery or digitised documents. If the user downloads a pdf file into our tool, the file is first converted into an image array with the help of the popular pdf2image python module. The pdf2image module is a wrapper around the pdftoppm and pdftocairo command line tools for converting PDFs into PIL image lists. The list of images shall be transmitted to the OCR. Each image is digitised to obtain readable text information. A combination of model recognition and characteristic detection is used to identify the letters, therefore, the words of the image. The identified text is sent back in string form.

Multimedia summarization has become a major requirement since Internet platforms such as YouTube provide easy access to enormous online resources. In general, automatic summarization aims at producing an abbreviated and informative version of its source. The kind of automatic summarization that we emphasise in this article is audio summarization, the source that corresponds to an audio signal. An audio summary can be done in the following three ways: directing the summary using only audio functions, extracting the text within the

audio signal, and directing the summarization process with textual methods, and a hybrid approach composed of a blend of the first two.

Each approach has its pros and cons. Using only audio features to create a summary has the advantage that it is completely independent from transcription; however, this can also be a problem since the abstract is based only on the way things are said. In contrast, leading the summary with textual methods benefits from the information contained in the text, dealing with more informative summaries; nevertheless, in some cases, transcripts are not available. Finally, the use of audio functions and textual methods can improve the quality of the summary; however, the disadvantages of both approaches exist.

Voice recognition is the interdisciplinary field that uses certain technologies to identify spoken language and translate it into text. The audio file downloaded by the user is initially pre-processed for transcription. An audio summary can be done in the following three ways: directing the summary using only audio functions, extracting the text within the audio signal, and directing the summarization process with textual methods, and a hybrid approach composed of a blend of the first two. Each approach has its pros and cons. Using only audio features to create a summary has the advantage that it is completely independent from transcription; however, this can also be a problem since the abstract is based only on the way things are said. In contrast, leading the summary with textual methods benefits from the information contained in the text, dealing with more informative summaries; nevertheless, in some cases, transcripts are not available. Finally, the use of audio functions and textual methods can improve the quality of the summary; however, the disadvantages of both approaches exist.

1.2 Working of Audio summarization

Audio summarization works by using different techniques to extract the essential information from an audio recording. Text summarization involves converting the audio into text, analysing the text, and selecting the most important information. Audio segmentation involves dividing the audio into smaller segments based on speaker change or content change. Speaker identification involves identifying different speakers in the audio and summarizing their contributions separately.

1.3 Applications of audio summarization

Audio summarization has numerous applications in various fields such as journalism, education, and business. In journalism, audio summarization can be used to quickly summarize interviews or press conferences. In education, audio summarization can be used to summarize lectures and make them more accessible to students. In business, audio summarization can be used to summarize meetings and presentations.

1.4 Real-time audio summarization

Real-time audio summarization involves summarizing the audio as it is being spoken. This can be achieved through the use of real-time speech recognition and summarization algorithms. Real-time audio summarization is particularly useful for live events such as conferences and speeches.

1.5 Tools and techniques for real-time audio summarization

Real-time audio summarization requires the use of specialized tools and

techniques. Speech recognition software such as Google Cloud Speech-to-Text, Microsoft Azure Speech Services, and Amazon Transcribe can be used to transcribe the audio in real-time. Summarization algorithms such as TextRank, LexRank, and LSA can be used to extract the most important information from the transcribed text.

1.6 Advantages of real-time audio summarization

Real-time audio summarization has numerous advantages over traditional audio summarization. It allows for the quick extraction and presentation of information, making it ideal for live events. It also reduces the need for manual transcription and summarization, saving time and effort.

1.7 Challenges of real-time audio summarization

Real-time audio summarization also has its challenges. The accuracy of speech recognition software can be affected by background noise, speaker accents, and technical issues. The summarization algorithms may also miss important information or include irrelevant information.

1.8 Impact of audio summarization on podcasting

Audio summarization has the potential to transform the podcasting industry. By providing quick and easy access to the most important information in a podcast episode, audio summarization can attract new listeners and improve the overall listening experience. It can also help podcasters to increase their reach and engagement with their audience.

1.9 Impact of audio summarization on audio books

Audio summarization can also have a significant impact on the audio book industry. By summarizing the essential information in an audio book, listeners can quickly decide whether or not the book is of interest to them. Audio summarization can also help listeners to revisit key points in the book without having to listen to the entire recording.

1.10 Future of audio summarization in real-time

The future of real-time audio summarization looks promising. Advancements in speech recognition technology and machine learning algorithms are improving the accuracy and effectiveness of real-time audio summarization. The integration of real-time audio summarization into virtual assistants and smart speakers also presents new opportunities for its use. As the demand for quick and easy access to information continues to grow, real-time audio summarization will become an increasingly important tool in various industries.

1.2 ORGANIZATION OF REPORT

The proposed work is divided into various chapters as mentioned below:
Chapter 1 is the introduction that explains the objective and scope of the proposed system. Chapter 2 is the literature survey that elaborates on the research works on the existing system and their issues. It also proposes a new system which makes an attempt on improving the existing system. Chapter 3 describes the system design and the roles played by various modules. Chapter 4 gives details about the module’s description. Chapter 5 provides the conclusion and timeline chart for PHASE II which summarizes the efforts undertaken in the proposed system and states the findings and various shortcomings in the proposed system.

1.3 PROBLEM DEFINITION

Audio summarization in real time for podcast, speeches, and audio books is a process that involves extracting the most important information from an audio recording as it is being spoken. The goal of audio summarization is to provide listeners with quick access to the essential information in the audio without having to listen to the entire recording. However, audio summarization in real-time poses several challenges. First, speech recognition software used to transcribe the audio in real-time can be affected by various factors such as background noise, speaker accents, and technical issues. This can lead to inaccuracies in the transcribed text, which can affect the quality of the summarized information. Second, summarization algorithms used to extract the most important information from the transcribed text may also face challenges. These algorithms may miss important information or include irrelevant information, leading to a less effective summary. Third, real-time audio summarization requires specialized tools and techniques, which may not be readily available or accessible to all users. This can limit the widespread adoption of real-time audio summarization technology. Finally, there are ethical and privacy concerns associated with the use of audio summarization technology. For example, the use of audio summarization technology in sensitive or private contexts, such as medical consultations or legal proceedings, may raise concerns about the protection of confidential information. Overall, these challenges highlight the need for further research and development in the field of real-time audio summarization. While real-time audio summarization offers numerous benefits, including faster access to information and improved listening experiences, addressing these challenges will be crucial to ensure the accuracy, effectiveness, and ethical use of the technology.

CHAPTER 2
LITERATURE REVIEWS

[1] A. Vartakavi, A. Garg and Z. Rafii, "Audio Summarization for Podcasts," 2021 29th European Signal Processing Conference (EUSIPCO), 2021

This paper proposes an audio summarization system for podcasts, which aims to provide an accurate and concise summary of the podcast content. The system uses a two-stage approach that first segments the audio stream into important chunks using an energy-based algorithm, and then generates a summary by selecting the most representative sentences from the segmented chunks using a sentence ranking method based on features such as speech rate, energy, and pitch. The paper evaluates the proposed system on a dataset of real podcasts and compares its performance with several baseline methods. The results show that the proposed system achieves high accuracy and efficiency, making it a promising solution for audio summarization of podcasts.

[2] H. Lee and G. Lee, "Hierarchical Model For Long-Length Video Summarization With Adversarially Enhanced Audio/Visual Features," 2020 IEEE International Conference on Image Processing (ICIP), 2020

This paper proposes a hierarchical model for long-length video summarization that leverages adversarially enhanced audio/visual features to provide an accurate and concise summary of the video content. The model uses a hierarchical structure that first segments the video into scenes, and then generates a summary by selecting the most representative frames from each scene using a novel adversarial learning approach that enhances the audio/visual features. The paper evaluates the proposed model on a dataset of real videos and compares its performance with several baseline methods. The results show that the proposed

model achieves high accuracy and efficiency, making it a promising solution for long-length video summarization.

[3] M. -H. Su, C. -H. Wu and H. -T. Cheng, "A Two-Stage Transformer-Based Approach for Variable-Length Abstractive Summarization," in IEEE/ACM Transactions on Audio, Speech, and Language Processing,2020.
This paper proposes a two-stage transformer-based approach for variable-length abstractive summarization, which aims to provide an accurate and concise summary of the input text of varying lengths. The first stage of the approach uses a transformer-based encoder-decoder model to generate an intermediate summary, which is then refined in the second stage using a transformer-based decoder model that generates a final summary. The paper evaluates the proposed approach on a dataset of real news articles and compares its performance with several baseline methods. The results show that the proposed approach achieves high accuracy and efficiency, making it a promising solution for variable-length abstractive summarization.

[4] Y. He, X. Xu, X. Liu, W. Ou and H. Lu, "Multimodal Transformer Networks with Latent Interaction for Audio-Visual Event Localization," 2021 IEEE International Conference on Multimedia and Expo (ICME), 2021
This paper proposes a multimodal transformer network with latent interaction for audio-visual event localization, which aims to accurately detect and localize events in videos using both audio and visual modalities. The proposed model uses a transformer-based architecture that processes the audio and visual features separately before fusing them using a latent interaction module. The paper evaluates the proposed model on a dataset of real videos and compares its performance with several baseline methods. The results show that the proposed model achieves high accuracy and efficiency, making it a promising solution for audio-visual event localization in videos.

[5] A. Gidiotis and G. Tsoumakas, "A Divide-and-Conquer Approach to the Summarization of Long Documents," in IEEE/ACM Transactions on Audio, Speech, and Language Processing,2020

This paper proposes a divide-and-conquer approach for the summarization of long documents, which aims to provide an accurate and concise summary of lengthy texts. The proposed approach uses a two-stage method that first divides the document into sections using a clustering algorithm, and then generates a summary by selecting the most representative sentences from each section using a sentence scoring method. The paper evaluates the proposed approach on a dataset of real documents and compares its performance with several baseline methods. The results show that the proposed approach achieves high accuracy and efficiency, making it a promising solution for the summarization of long documents.

[6] R. K. Yadav, R. Bharti, R. Nagar and S. Kumar, "A Model For Recapitulating Audio Messages Using Machine Learning," 2020 International Conference for Emerging Technology (INCET), 2020

This paper proposes a machine learning-based model for recapitulating audio messages, which aims to provide an accurate and concise summary of audio messages. The proposed model uses a combination of feature extraction techniques and machine learning algorithms to identify important segments of the audio and generate a summary. The paper evaluates the proposed model on a dataset of real audio messages and compares its performance with several baseline methods. The results show that the proposed model achieves high accuracy and efficiency, making it a promising solution for recapitulating audio messages using machine learning.

[7] H. Zhu, L. Dong, F. Wei, B. Qin and T. Liu, "Transforming Wikipedia Into Augmented Data for Query-Focused Summarization," in IEEE/ACM Transactions on Audio, Speech, and Language Processing,2022

This paper proposes a method for transforming Wikipedia into augmented data for query-focused summarization, which aims to provide a summary of a long document that is relevant to a given query. The proposed method uses a transformer-based model to generate query-focused summaries, and leverages Wikipedia as a source of augmented data to improve the performance of the summarization model. The paper evaluates the proposed method on a dataset of real documents and compares its performance with several baseline methods. The results show that the proposed method achieves high accuracy and efficiency, making it a promising solution for query-focused summarization using augmented data from Wikipedia.

[8] P. G. Shambharkar and R. Goel, "Analysis of Real Time Video Summarization using Subtitles," 2021 International Conference on Industrial Electronics Research and Applications (ICIERA), 2021

This paper proposes a method for real-time video summarization using subtitles, which aims to provide an accurate and concise summary of a video in real-time. The proposed method uses a combination of natural language processing and computer vision techniques to extract the most important information from both the audio and subtitle tracks of the video. The paper evaluates the proposed method on a dataset of real videos and compares its performance with several baseline methods. The results show that the proposed method achieves high accuracy and efficiency, making it a promising solution for real-time video summarization using subtitles.

[9] P. Gupta, S. Nigam and R. Singh, "A Ranking based Language Model for Automatic Extractive Text Summarization," 2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR), 2022

This paper proposes a ranking-based language model for automatic extractive text summarization, which aims to provide a concise summary of a long document by selecting the most important sentences from the original text. The proposed model uses a ranking algorithm that scores each sentence based on several features, including word frequency, sentence position, and sentence length. The paper evaluates the proposed model on a dataset of real documents and compares its performance with several baseline methods. The results show that the proposed model achieves high accuracy and efficiency, making it a promising solution for automatic extractive text summarization**.**

[10] B. Zhao, M. Gong and X. Li, "AudioVisual Video Summarization," in IEEE Transactions on Neural Networks and Learning

This paper proposes an audiovisual video summarization method, which aims to provide an accurate and concise summary of a long video by leveraging both audio and visual information. The proposed method uses a multi-modal deep neural network that processes the audio and visual features separately before fusing them in a joint embedding space. The paper evaluates the proposed method on a dataset of real videos and compares its performance with several baseline methods. The results show that the proposed method achieves high accuracy and efficiency, making it a promising solution for audiovisual video summarization.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system only identifies emotions when any audio input is given. The existing system uses python speech recognition library which might not work for all the inputs since the library has dependency issues. The existing system can’t analyze if a big dataset of audio files is passed as an input because the existing system isn't trained with large dataset.

Several existing systems offer audio summarization in real-time for podcasts, speeches, and audio books. These systems use various techniques and tools to transcribe the audio, segment the text, and summarize the content. Here are some examples of existing systems:

Amazon Transcribe: Amazon Transcribe is a cloud-based automatic speech recognition service that can transcribe audio in real-time. It can identify different speakers and transcribe the audio with high accuracy. The transcribed text can be used to summarize the audio recording.

Google Cloud Speech-to-Text: Google Cloud Speech-to-Text is another cloud-based speech recognition service that can transcribe audio in real-time. It uses advanced machine learning algorithms to identify and transcribe speech accurately.

Otter.ai: Otter.ai is a speech recognition and transcription tool that uses AI-powered algorithms to transcribe audio in real-time. It can identify different speakers and provides real-time transcription and summarization.

Podchaser: Podchaser is a podcast database that offers a feature called "Quick Picks" which provides a short summary of podcast episodes. This feature is generated using NLP algorithms that analyze the podcast transcript and identify key phrases.

Sonix.ai: Sonix.ai is a web-based transcription and translation service that offers real-time audio summarization. It uses advanced NLP algorithms to analyze the transcribed text and identify key phrases.

These systems use a combination of speech recognition, natural language processing, and machine learning algorithms to provide real-time audio summarization. They are designed to handle various challenges such as speaker identification, background noise, and accents. However, they still face challenges such as accuracy and reliability, and the need for continuous improvement in order to provide a more effective and reliable audio summarization in real-time.

3.2 PROPOSED SYSTEM

To begin, we will have to register for an account in order to retrieve the data and sign up for the free plan in order to make use of the Listen Notes API. With this plan, you have the ability to have a maximum of 300 requests processed each month, which may be sufficient for working on personal projects. The next step is to navigate to the Listen Notes page for the podcast, click the episode in which we are interested, and then select the option to "Use API to get this episode." An request for transcription will be sent. We are now sending a POST request to the transcript endpoint of the Assembly AI rather than making a GET request like we were doing previously. When uploading the audio URL to Assembly AI, the post method is the one that is used. This step is essential in order to acquire the

transcription as well as the summary, both of which will be produced if we configure auto chapter to have a value of True. We would only be able to retrieve the transcription if we set auto chapter to the value False. The id of the transcription answer is then saved after this step. Finally, we can obtain the transcription and the summary by sending a GET request to Assembly AI. We need to make few requests until the status of the response is completed. Later, we save the results into two different files, a txt file for the transcription and a JSON file for the summary. The process of translating an audio file into a text file is referred to as audio transcription. This could be an audio recording of anything; for example, an interview, an academic study, a music video clip, or a recording of a conference.

A proposed system for audio summarization in real-time for podcast, speeches, and audio books would involve the following components and techniques:

Audio Input: The system would allow for the input of high-quality audio recordings from different sources, including live streams, recorded files, and microphone input.

Speech Recognition: The system would use advanced speech recognition software to transcribe the audio input into text in real-time. The software would be optimized to recognize different accents, dialects, and background noise to ensure high accuracy in the transcription.

Segmentation: The system would segment the transcribed text into smaller units based on speaker change or content change. This segmentation would be done

automatically by the system, based on predetermined criteria such as silence detection or speaker identification.

Summarization: The system would use advanced natural language processing algorithms to analyze the segmented text and identify the most important information. This would involve techniques such as topic modeling, sentiment analysis, and keyword extraction to identify the key concepts and themes in each segment.

Output: The system would output the summarized text in real-time, using a user-friendly interface that presents the information in an easily digestible format. The output could be in the form of bullet points, summaries, or highlights, and would allow users to quickly access the most important information in the audio recording.

The proposed system would leverage the latest advances in speech recognition, natural language processing, and machine learning to provide highly accurate and effective real-time audio summarization. The system would also be customizable, allowing users to adjust the settings and criteria used for segmentation and summarization based on their specific needs and preferences.

Overall, the proposed system would be a powerful tool for podcasters, journalists, educators, and other professionals who need to quickly access essential information in audio recordings. It would streamline the process of summarizing audio recordings and make it more accessible and efficient, saving time and effort for users.

3.3 FEASIBILITY STUDY

With an eye towards gauging the project's viability and improving server performance, a business proposal defining the project's primary goals and offering some preliminary cost estimates is offered here. Your proposed system's viability may be assessed once a comprehensive study has been performed. It is essential to have a thorough understanding of the core requirements of the system at hand before beginning the feasibility study. The feasibility research includes mostly three lines of thought:

- Economic feasibility
- Technical feasibility
- Operational feasibility
- Social feasibility

3.3.1 ECONOMICAL FEASIBILITY

The study's findings might help upper management estimate the potential cost savings from using this technology. The corporation can only devote so much resources to developing and analyzing the system before running out of money. Every dollar spent must have a valid reason. As the bulk of the used technologies are open-source and free, the cost of the updated infrastructure came in far cheaper than anticipated. It was really crucial to only buy customizable products.

3.3.2 TECHNICAL FEASIBILITY

This research aims to establish the system's technical feasibility to ensure its smooth development. Adding additional systems shouldn't put too much pressure

on the IT staff. Hence, the buyer will experience unnecessary anxiety. Due to the low likelihood of any adjustments being necessary during installation, it is critical that the system be as simple as possible in its design.

3.3.3 OPERATIONAL FEASIBILITY

An important aspect of our research is hearing from people who have actually used this technology. The procedure includes instructing the user on how to make optimal use of the resource at hand. The user shouldn't feel threatened by the system, but should instead see it as a necessary evil. Training and orienting new users has a direct impact on how quickly they adopt a system. Users need to have greater faith in the system before they can submit constructive feedback.

3.3.4 SOCIAL FEASIBILITY

During the social feasibility analysis, we look at how the project could change the community. This is done to gauge the level of public interest in the endeavour. Because of established cultural norms and institutional frameworks, it's likely that a certain kind of worker will be in low supply or nonexistent.

3.4 REQUIREMENT SPECIFICATION

3.4.1 HARDWARE REQUIREMENTS

Processor	: Pentium Dual Core 2.00GHZ
Hard disk	: 120 GB
RAM	: 2GB (minimum)
Keyboard	: 110 keys enhanced

3.4.2 SOFTWARE REQUIREMENTS

Operating system	: Windows7 (with service pack 1), 8, 8.1 and 10
Tool	: Anaconda with Jupyter Notebook
Language	: Python

3.5 LANGUAGE SPECIFICATION– PYTHON

Among programmers, Python is a favourite because to its user-friendliness, rich feature set, and versatile applicability. Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively utilised by the programming community.

Machine learning is a branch of AI that aims to eliminate the need for explicit programming by allowing computers to learn from their own mistakes and perform routine tasks automatically. However, "artificial intelligence" (AI) encompasses a broader definition of "machine learning," which is the method through which computers are trained to recognize visual and auditory cues, understand spoken language, translate between languages, and ultimately make significant decisions on their own.

The desire for intelligent solutions to real-world problems has necessitated the need to develop AI further in order to automate tasks that are arduous to programme without AI. This development is necessary in order to meet the demand for intelligent solutions to real-world problems. Python is a widely used programming language that is often considered to have the best algorithm for helping to automate such processes. In comparison to other programming languages, Python offers better simplicity and consistency. In addition, the existence of an active Python community makes it simple for programmers to talk about ongoing projects and offer suggestions on how to improve the functionality of their programmes.

ADVANTAGES OF USING PYTHON

Following are the advantages of using Python:

- **Variety of Framework and libraries:**

A good programming environment requires libraries and frameworks. Python frameworks and libraries simplify programme development. Developers can speed up complex project coding with prewritten code from a library. PyBrain, a modular machine learning toolkit in Python, provides easy-to-use algorithms. Python frameworks and libraries provide a structured and tested environment for the best coding solutions.

- **Reliability**

Most software developers seek simplicity and consistency in Python. Python code is concise and readable, simplifying presentation. Compared to other programming languages, developers can write code quickly. Developers can get community feedback to improve their product or app. Python is simpler than other programming languages, therefore beginners may learn it quickly. Experienced developers may focus on innovation and solving real-world

problems with machine learning because they can easily design stable and trustworthy solutions.

- **Easily Executable**

Developers choose Python because it works on many platforms without change. Python runs unmodified on Windows, Linux, and macOS. Python is supported on all these platforms; therefore, you don't need a Python expert to comprehend it. Python's great executability allows separate applications. Programming the app requires only Python. Developers benefit from this because some programming languages require others to complete the job. Python's portability cuts project execution time and effort.

CHAPTER 4

SYSTEM DESIGN

4.1 ER DIAGRAM

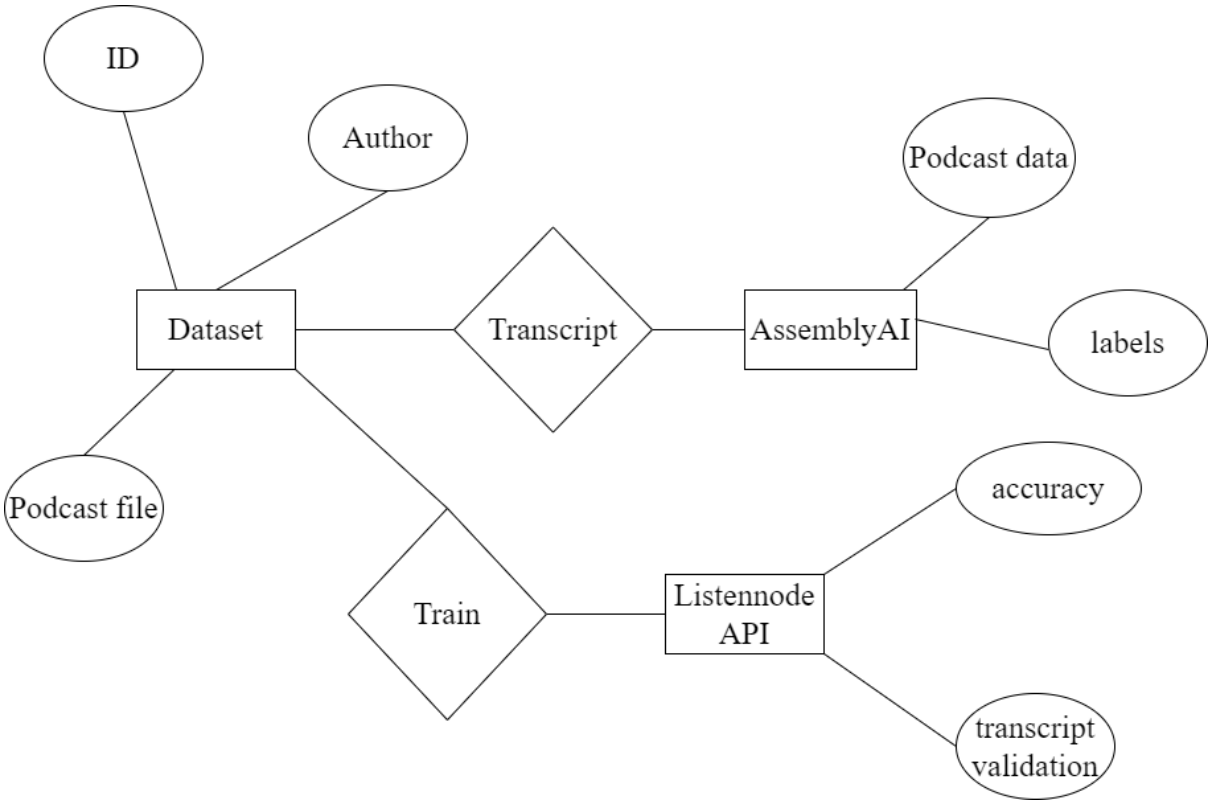


Fig 4.1 – ER Diagram

Description:

The abbreviation ER refers to a connection between two entities. The entities used and saved in the database are shown in relationship diagrams. They break down the process into its component parts and explain how they work. Attributed concepts, Relationship concepts, and Entity concepts are the building blocks for these kinds of diagrams.

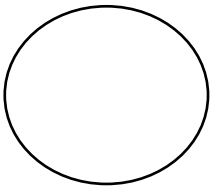
4.2 DATAFLOW DIAGRAM

To illustrate the movement of information throughout a procedure or system, one might use a Data-Flow Diagram (DFD). A data-flow diagram does not include any decision rules or loops, as the flow of information is entirely one-way. A flowchart can be used to illustrate the steps used to accomplish a certain data-driven task. Several different notations exist for representing data-flow graphs. Each data flow must have a process that acts as either the source or the target of the information exchange. Rather than utilizing a data-flow diagram, users of UML often substitute an activity diagram. In the realm of data-flow plans, site-oriented data-flow plans are a subset. Identical nodes in a data-flow diagram and a Petri net can be thought of as inverted counterparts since the semantics of data memory are represented by the locations in the network. Structured data modeling (DFM) includes processes, flows, storage, and terminators.

Data Flow Diagram Symbols

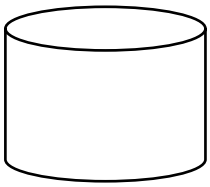
Process

A process is one that takes in data as input and returns results as output.



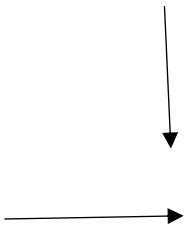
Data Store

In the context of a computer system, the term "data stores" is used to describe the various memory regions where data can be found. In other cases, "files" might stand in for data.



Data Flow

Data flows are the pathways that information takes to get from one place to another. Please describe the nature of the data being conveyed by each arrow.



External Entity

In this context, "external entity" refers to anything outside the system with which the system has some kind of interaction. These are the starting and finishing positions for inputs and outputs, respectively.

DATA FLOW DIAGRAM

The whole system is shown as a single process in a level DFD. Each step in the system's assembly process, including all intermediate steps, are recorded here. The "basic system model" consists of this and 2-level data flow diagrams.

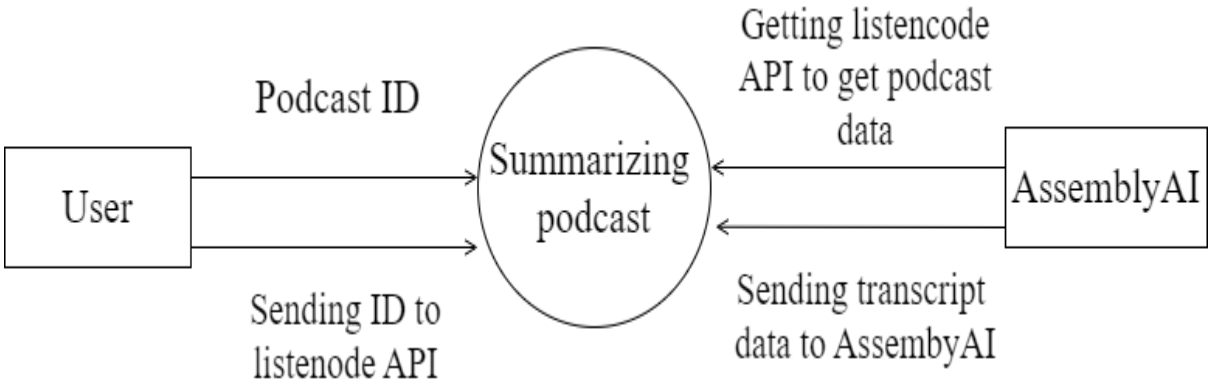


Fig 4.2 – Data Flow Level 0 Diagram

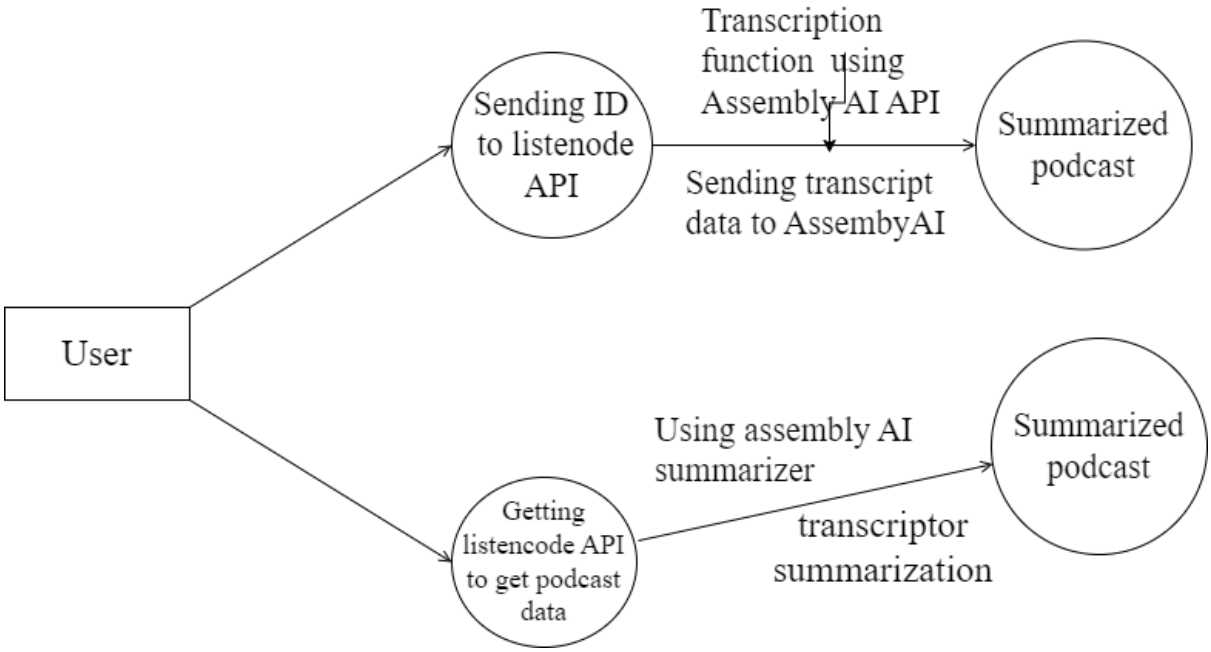


Fig 4.3 – Data Flow Level 1 Diagram

4.3 UML DIAGRAMS

4.3.1 USE CASE DIAGRAM

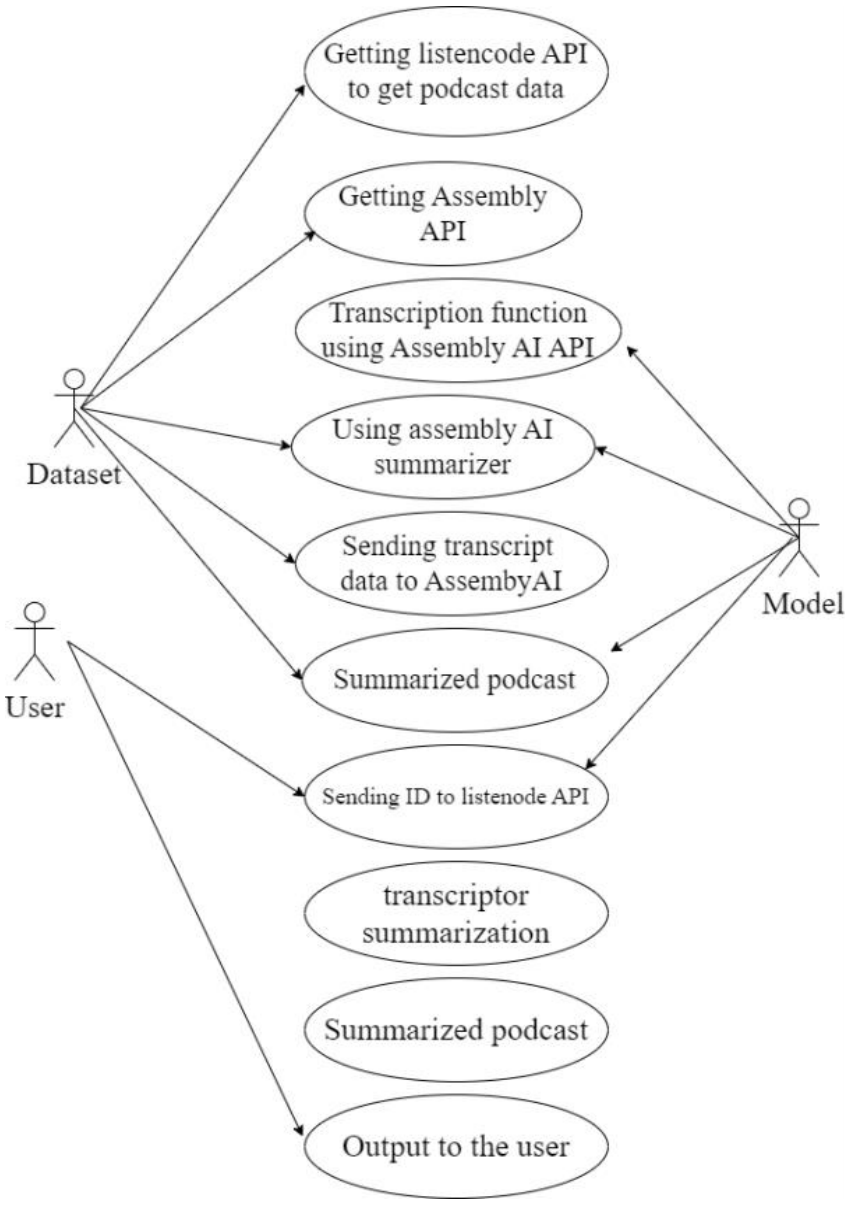


Fig 4.4 Use case Diagram

Description:

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a

system and also tells how the user handles a system. The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

4.3.2 CLASS DIAGRAM

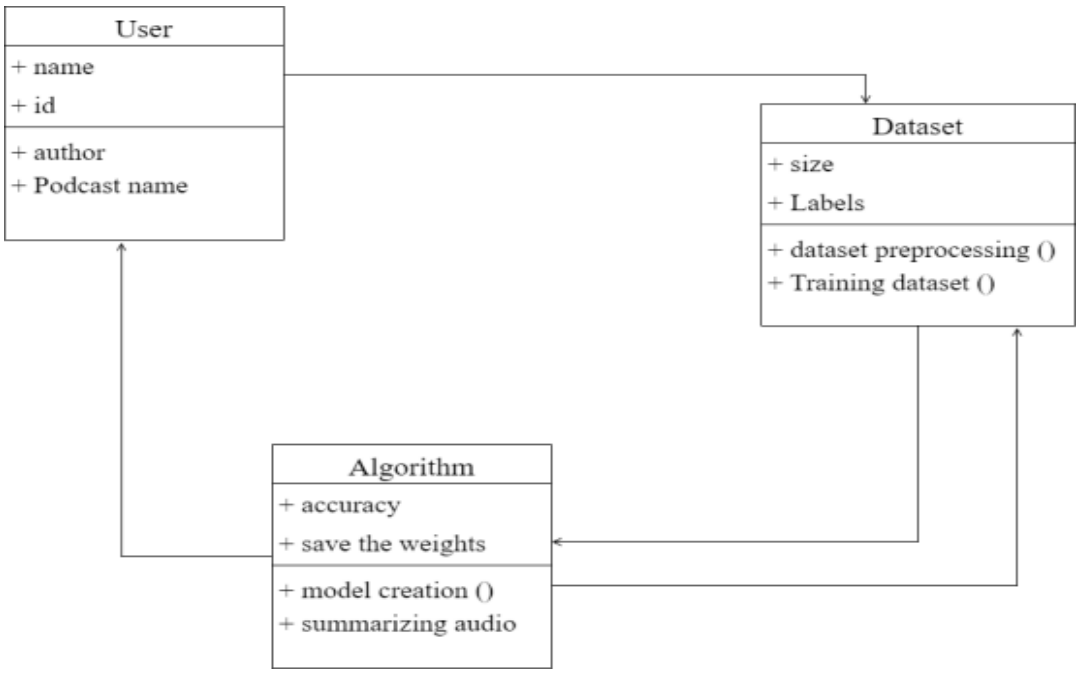


Fig 4.5 Class Diagram

Description:

A static view of an application is depicted in the class diagram. It displays the properties, classes, functions, and relationships of the software system to provide an overview of the software system. It organizes class names, properties, and functions into a discrete compartment to aid in program development.

The following are the functions of class diagrams:

1. Define the main responsibilities of the system.

2. Serves as a basis for component diagrams and deployment. Use forward and reverse engineering.

4.3.3 SEQUENCE DIAGRAM

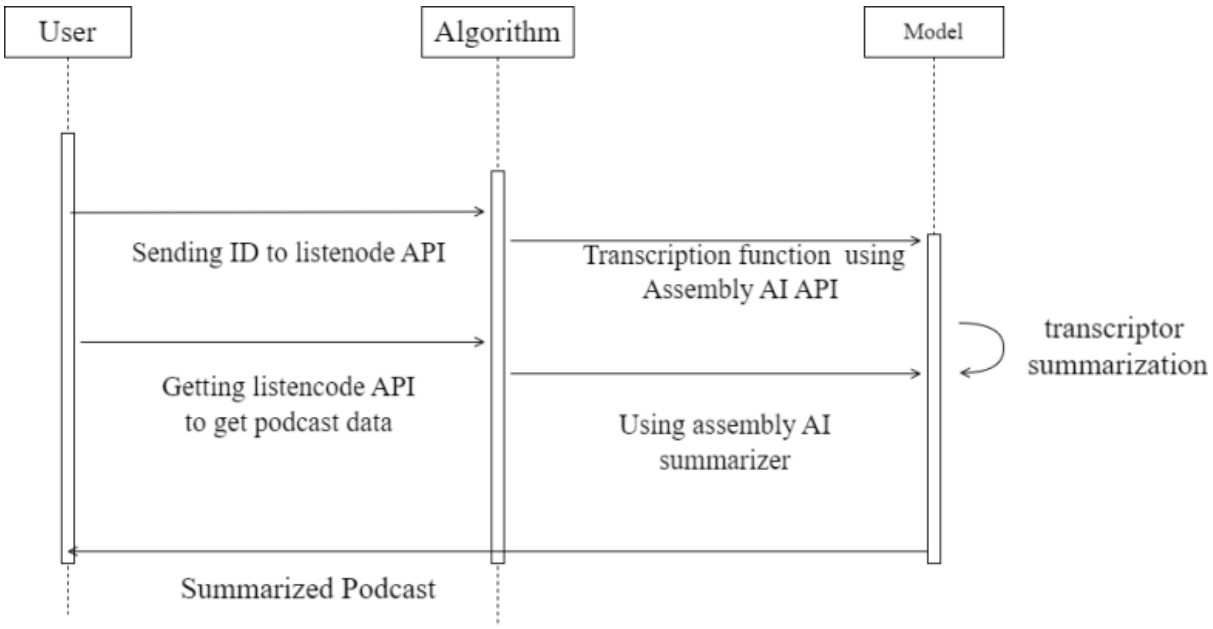


Fig 4.6 Sequence Diagram

Description:

The sequence diagram, also called the event diagram, describes the flow of messages in the system. It helps to visualize various dynamic parameters. He describes the communication between two rescue lines as a series of events arranged in time in which these rescue lines participated during the performance. The lifeline is represented by a vertical bar in UML while the message flow is represented by a vertical dotted line that crosses the bottom of the page. It includes both repetitions and branches.

4.3.4 ACTIVITY DIAGRAM

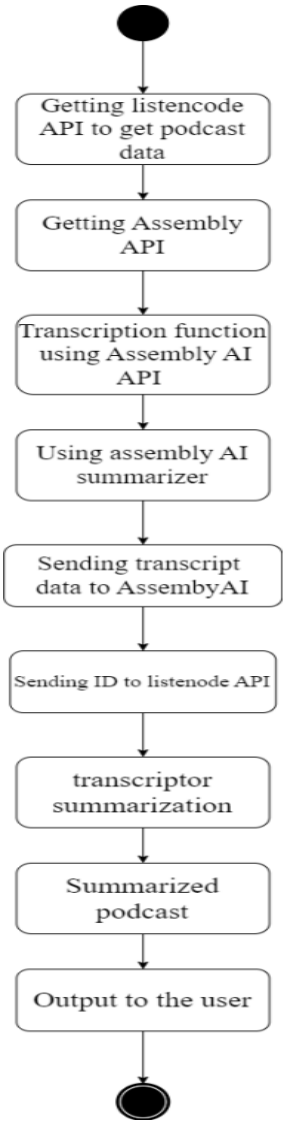


Fig 4.7 Activity Diagram

Description:

An activity diagram is a kind of graphical representation that may be used to depict events visually. It is made up of a group of nodes that are linked to one another by means of edges. They are able to be connected to any other modelling element, which enables the behaviour of activities to be replicated using that methodology. Simulations of use cases, classes, and interfaces, as well as component collaborations and component interactions, are all made feasible with the help of this tool.

CHAPTER 5
SYSTEM ARCHITECTURE

This graphic provides a concise and understandable description of all the entities currently integrated into the system. The diagram shows how the many actions and choices are linked together. You might say that the whole process and how it was carried out is a picture. The figure below shows the functional connections between various entities.

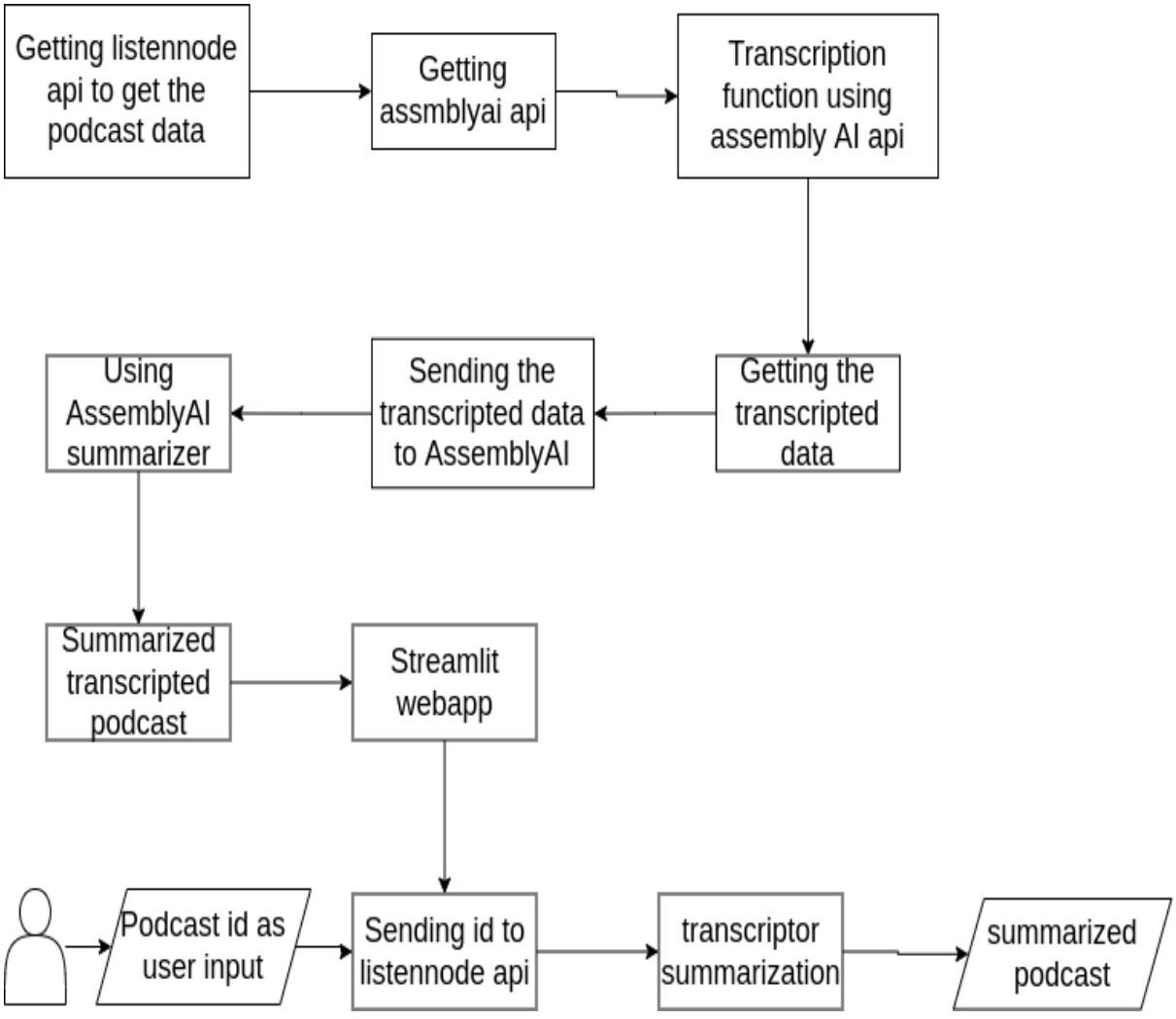


Fig 5.1 – Architecture Diagram

5.1 MODULE DESIGN SPECIFICATION

The recent meteoric rise in popularity of podcasts poses both a massive potential and an entirely new set of problems to the systems that are currently in place for content discovery and recommendation. In contrast to listening to music, listening to a podcast typically requires the listener's full and undivided attention for extended periods of time. The listener's preference may be influenced by subjective characteristics, such as the speaker's presentation style, the sort of humour used, or the production quality; yet, it is difficult to determine these characteristics from a text description.

In the realm of video, movie trailers provide viewers with the opportunity to get a sneak peek at some of the film's content and then make an informed decision about whether or not to watch the movie. Because podcast episodes are typically released on a regular basis, it would be impossible to create trailers for each new episode individually. It has been demonstrated that the performance of spoken document search algorithms can be improved with the use of audio summaries.

The creation of brief audio summaries of podcasts in an automatic fashion is the focus of our latest method proposal. These kinds of summaries have the potential to educate the listener not only about the subject matter of the podcast but also about its subjective aspects, such as the presenting style and the level of production quality. An audio-based summarising algorithm faces a special set of obstacles while attempting to summarise podcasts. Podcasts, for instance, typically centre their attention on content that is conveyed through spoken word and frequently include overlapping speech from many speakers, free-form speech, audio effects, background music, and commercials.

In order to properly evaluate the significance of an audio segment, a supervised learning system that operates in the audio domain would first need to determine the nature of the audio segment in question. This would require a significant amount of training data to be annotated by hand, which is a procedure that is both

challenging and time-consuming because it involves listening to the audio in numerous passes. On the other hand, despite the fact that podcasts predominantly feature content in the form of spoken words, summarising can also be conducted in the text-domain on the episode's transcript.

There has only been a little amount of study done on automatic algorithms for summarising the audio of podcasts. The variety and narrative quality of podcasts, which sometimes include impromptu speech as well as music, sound effects, and commercials, may offer difficulties for the speech summary algorithms that are currently in use. In order to solve this challenge, we approach the problem of podcast audio summarization as a multi-modal data summarization problem. Specifically, we produce an audio summary of a podcast by using the text-domain as a guide.

We are going to transcribe and summarise one episode from listennote.com as part of this project, which is a podcast that discusses a variety of data science issues and offers advice on how to become a data scientist. The episode will be built as a web application.

Streamlit will be used in the construction of the application, and the AssemblyAI API will enable us to transcribe and summarise the episode of our podcast. This application programming interface (API) will first use automatic speech recognition to turn the audio into text, and then it will provide a summary of the text.

5.1.1 MODULE 1: Getting the PODCAST from the listennote api

We will create a method that will get the podcast data from the podcast id given as user input.

Extract Episode's URL from Listen Notes

Listen Notes is a robust online search engine and database for podcasts. It offers an API to gain access to podcast data and makes it possible to create new services and applications that are based on it. During the course of this section, we are going to extract the episode's url from our target podcast by making use of the Listen Notes API.

- To begin, we will have to register for an account in order to retrieve the data and sign up for the free plan in order to make use of the Listen Notes API. With this plan, you have the ability to have a maximum of 300 requests processed each month, which may be sufficient for working on personal projects.
- The next step is to navigate to the Listen Notes page for the podcast, click the episode in which we are interested, and then select the option to "Use API to get this episode.
- Following that, we will be able to modify the language code to Python and select requests from the list of available options so that we can utilize the library later.
- Make sure to insert the code into your notebook or script after you've copied it.

The retrieval of information from the Listen Notes API is being accomplished through the use of a tactical operation known as submitting a GET request to the endpoint of the Listen Notes Podcast API. After everything is said and done, we

store the final result as a JSON object. This object contains the episode's URL, which will be required at a later time.

In addition to that, we are importing a JSON file with the name secrets.json. This file is quite similar to a dictionary in that it contains a collection of key-value pairs much like a dictionary does. It is necessary for it to hold the API keys for AssemblyAI and Listen Notes, respectively. You will need to log into your accounts to access them.

Listennode API

Listen Node API is where we can search and get data related to API using podcast id.

```
api_key = os.environ.get("LISTEN_API_KEY", None)
client = podcast_api.Client(api_key=api_key)
```

5.1.2 MODULE 2: Transcription and summarization of the podcast

In this section, a request for transcription will be sent. We are now sending a POST request to the transcript endpoint of the AssemblyAI rather than making a GET request like we were doing previously. When uploading the audio URL to Assembly AI, the post method is the one that is used.

This step is essential in order to acquire the transcription as well as the summary, both of which will be produced if we configure the auto chapter to have a value of True. We would only be able to retrieve the transcription if we set the auto chapter to the value False. The id of the transcription answer is then saved after this step.

Retrieve Transcription and Summary:

Finally, we can obtain the transcription and the summary by sending a GET request to Assembly AI. We need to make a few requests until the status of the

response is completed. Later, we save the results into two different files, a txt file for the transcription and a JSON file for the summary.

Transcription:

The process of translating an audio file into a text file is referred to as audio transcription. This could be an audio recording of anything; for example, an interview, an academic study, a music video clip, or a recording of a conference. There are a lot of different situations in which having an audio recording rather than a written file is more convenient to have.

We will make use of the AssemblyAI API to provide the data that was fetched by the listennode API, and as a result, we will receive the data that was transcribed from the API.

AssemblyAI API:

Accurate text transcriptions can be generated from real-time audio streams as well as previously recorded audio files with the usage of the AssemblyAI Application Programming Interface (API).

In addition to transcription, the API provides a set of capabilities known as Audio Intelligence that may be used to better comprehend the audio data you have. These functions include Sentiment Analysis, Summarization, Entity Detection, Topic Detection, and many more.

Asynchronous Transcription:

The term "asynchronous transcription" refers to the process of transcribing audio or video data that have already been captured.

When we provide an audio file to be transcribed, it will take between 15 and 30 percent of the time that the audio file takes to complete. For instance, the completion of a file that takes 10 minutes would take about 1.5 minutes, but it may take up to 3 minutes.

Real-Time Streaming Transcription:

Real-Time Streaming WebSocket API streams text transcriptions back to clients within a few hundred milliseconds.

Errors Making Requests to the API

The API will always return a JSON response when there is an error.

Invalid API Token

API queries that are executed with an invalid API token will invariably result in the return of a status code of 401.

If a transcription job is unsuccessful, the status of the transcription will change to error, and an error key will be included in the JSON response that is sent by the API whenever the transcription is retrieved using the GET request. In addition, the status of the transcription will change to error if the job fails to be transcribed. In most cases, a failed transcript can be attributed to any one of the following causes:

- Format of audio files that is not supported
- Audio file did not contain audio data
- The length of the audio file was insufficient (less than 200 milliseconds).
- The URL of the audio file cannot be accessed.
- An mistake on our side

Resubmitting a file for transcription is something we always recommend doing in the event that a transcription task fails due to an error on our end (a server fault, for example). If you resubmit the file, in most cases a different server in our cluster will be able to correctly process your audio file.

Summarization

A shortened version of an audiobook is referred to as an audio summary. These summaries do an excellent job at condensing the most important ideas and themes from longer audiobooks into a style that is easier to understand and consume. They do an excellent job at capturing the author's work in all of its facets, including the content, the style, and the spirit.

We will transmit the data that has been transcribed over the same AssemblyAI API that will be used to retrieve the data that has been summarised from the podcast transcription using the AssemblyAI API.

5.1.3 MODULE 3: Web App

We are going to construct a webapp with Streamlit that will ask the user for the ID of the podcast they want to listen to and then provide them a transcripted summary of the podcast. The objective is to design an application utilising Streamlit, which is a free open-source framework that enables the construction of apps in Python with a minimal amount of lines of code.

To begin, all that needs to be done is to import the Python libraries and declare the functions that will replicate the actions that were just presented. In addition to that, we will also provide a zipped package that will contain both the file containing the transcription and the file containing the summary.

At this point, we are able to specify the primary components of our project. To begin, we make use of st.markdown to display the primary title of the application. When we have finished, we will insert the episode id of our postcast into the left panel sidebar that we created using the st.sidebar function. After you have added the id, you will need to click the "Submit" button.

When the button is pressed, the application will carry out all of the procedures necessary to transcribe and summarise the audio of the episode. After an interval of a few minutes, the outcomes will be displayed on the website. In the event that we would want to download the results, there is a button labelled Download that

will enable you to compress the transcription as well as the summary into a single file. Both the local URL and the Network URL are automatically returned by the system. Simply selecting one of these links will result in the desired outcome. Now, there is a fantastic app available that can transcribe and summarise your favourite podcast for you!

5.2 ALGORITHMS

Audio summarization in real time for podcasts, speeches, and audio books is a complex task that involves several steps. The algorithm starts by receiving the audio file as input and converting it into a digital format, such as MP3, WAV, or FLAC. Then, a speech-to-text API or software is used to transcribe the audio into text. The text data is pre-processed to remove stop words, punctuation, and other noise. Natural language processing (NLP) techniques are used to extract key sentences and phrases from the text, and scores are assigned to each sentence based on their relevance to the overall topic of the audio. Next, a clustering algorithm is used to group the top-scoring sentences into topics. For each topic, the most representative sentence is selected to serve as the topic summary. Finally, the topic summaries are combined to create the final audio summary. Text-to-speech software can be used to generate an audio summary from the selected sentences if desired. To achieve real-time processing, the algorithm needs to be optimized for speed and efficiency. This could involve using a stream-based approach to process the audio and text data in real time or using parallel processing to speed up computation. The accuracy of the transcription and NLP techniques used is also crucial to producing a high-quality summary. The algorithm can be implemented on a powerful server or cloud platform to handle the processing load.

Audio summarization in real-time for podcast, speeches, and audio books is a complex process that involves several stages. A generalized algorithm for this process can be broken down into the following steps:

Step 1: Audio Input

The first step in the audio summarization process is to input the audio recording into the system in real-time. This involves capturing the audio using a microphone or accessing an audio file that is being streamed or downloaded in real-time. The audio input should be of high quality and without any background noise or distortion.

Step 2: Speech Recognition

Once the audio input is captured, the next step is to transcribe the audio into text. This is done using speech recognition software, which converts the audio signal into a digital format and applies machine learning algorithms to identify and transcribe spoken words. The accuracy of speech recognition software can be affected by various factors, such as background noise, speaker accents, and technical issues. Therefore, it is essential to use high-quality speech recognition software that is trained to recognize various accents and can filter out background noise.

Step 3: Segmentation

After the audio is transcribed into text, the next step is to segment the text into smaller units based on speaker change or content change. This is done to identify different topics or speakers within the audio recording. For instance, in a podcast, different segments can be identified based on the introduction, guest interview,

and conclusion. In speeches, different segments can be identified based on the speaker's introduction, key points, and conclusion. In audio books, different segments can be identified based on the chapter titles and subheadings. Segmenting the text can help to identify the most important parts of the audio recording and focus the summarization process on these segments.

Step 4: Summarization

Once the text is segmented, the next step is to summarize the content of each segment. This is done using text summarization algorithms that identify the most important information within each segment. There are several techniques used for text summarization, such as frequency-based approaches, graph-based approaches, and machine learning-based approaches. Frequency-based approaches rank sentences based on their occurrence frequency within the segment, while graph-based approaches use a network of words to identify the most important concepts within the text. Machine learning-based approaches use supervised or unsupervised machine learning algorithms to identify the most important sentences or concepts within the text. The output of the summarization step is a summary of each segment that contains the most important information.

Step 5: Output

The final step is to output the summarized text in real-time. The summarized text can be presented in different formats, such as bullet points, headlines, or short paragraphs. The output can be displayed on a screen, read aloud by a text-to-speech engine, or sent to a user's device for later access. Real-time output allows users to quickly access the essential information in the audio recording without having to listen to the entire recording.

The above steps constitute a generalized algorithm for audio summarization in real-time for podcasts, speeches, and audio books. The algorithm can be further optimized and customized based on the specific needs and requirements of each application. Real-time audio summarization has the potential to transform the way we consume and interact with audio content, and further research and development in this area can lead to significant advancements in the field of natural language processing and speech recognition.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 CODING

main.py

```
import streamlit as st
import requests
import zipfile
import json
from time import sleep


def retrieve_url_podcast(parameters,episode_id):
    url_episodes_endpoint = 'https://listen-
api.listennotes.com/api/v2/episodes'
    headers = {
    'X-ListenAPI-Key': parameters["api_key_listennotes"],
    }
    url = f"{url_episodes_endpoint}/{episode_id}"
    response = requests.request('GET', url, headers=headers)
    # print(response.json())
    data = response.json()
    audio_url = data['audio']
    return audio_url
```

```
def send_transc_request(headers,api_key,audio_url):
    transcript_endpoint = "https://api.assemblyai.com/v2/transcript"
    transcript_request = {
            'audio_url': audio_url,
            'auto_chapters': True
        }
    transcript_response = requests.post(transcript_endpoint,
json=transcript_request, headers=headers)
    transcript_id = transcript_response.json()["id"]
    # print(transcript_id)
    return transcript_id


def obtain_polling_response(headers,transcript_id):
    polling_endpoint =
f"https://api.assemblyai.com/v2/transcript/{transcript_id}"
    polling_response = requests.get(polling_endpoint, headers=headers)
    i=0
    while polling_response.json()["status"] != 'completed':
        sleep(5)
        polling_response = requests.get(polling_endpoint, headers=headers)
```

```
    return polling_response


def save_files(transcription,summary):


    with open('transcript.txt', 'w') as f:
        f.write(transcription)
        f.close()
    with open('summary.txt', 'w') as f:
        f.write(summary)
        f.close()
    list_files = ['transcript.txt','summary.txt']
    with zipfile.ZipFile('final.zip', 'w') as zipF:
      for file in list_files:
         zipF.write(file, compress_type=zipfile.ZIP_DEFLATED)
      zipF.close()


# title of web app


st.markdown('# **Summarizer App**')
bar = st.progress(0)


st.sidebar.header('Input parameter')
```

```
with st.sidebar.form(key='my_form'):
    episode_id = st.text_input('Insert Episode ID:')
    #bbc965b98747439abf0fe5c1a5ddfe5c
    #e9baa9118e654cd09baff7ac4b67228f
    submit_button = st.form_submit_button(label='Submit')


if submit_button:
    f = open("secrets.json", "rb")
    parameters = json.load(f)
    # step 1 - Extract episode's url from listen notes
    audio_url = retrieve_url_podcast(parameters,episode_id)
    #bar.progress(30)
    api_key = parameters["api_key"]
    headers = {
        "authorization": api_key,
        "content-type": "application/json"
    }
    # step 2 - retrieve id of transcription response from AssemblyAI
    transcript_id = send_transc_request(headers,api_key,audio_url)
    #bar.progress(70)
    # step 3 - transcription and summary
```

```
polling_response = obtain_polling_response(headers,transcript_id)
transcription = polling_response.json()["text"]
print(transcription)
chapters = polling_response.json()['chapters']
print(chapters)
summary = [c['summary'] for c in chapters]
summary = '. '.join(summary)

#bar.progress(100)
st.header('Transcription')
st.success(transcription)
st.header('Summary')
st.success(summary)
save_files(transcription,summary)

with open("final.zip", "rb") as zip_download:
    btn = st.download_button(
        label="Download",
        data=zip_download,
        file_name="final.zip",
        mime="application/zip"
    )
```

summary.py

```
import streamlit as st
import requests
import zipfile
import json
from time import sleep

def retrieve_url_podcast(parameters,episode_id):
    url_episodes_endpoint = 'https://listen-api.listennotes.com/api/v2/episodes'
    headers = {
    'X-ListenAPI-Key': parameters["api_key_listennotes"],
    }
    url = f"{url_episodes_endpoint}/{episode_id}"
    response = requests.request('GET', url, headers=headers)
    # print(response.json())
    data = response.json()
```

```
    audio_url = data['audio']
    return audio_url


def send_transc_request(headers,api_key,audio_url):
    transcript_endpoint = "https://api.assemblyai.com/v2/transcript"
    transcript_request = {
            'audio_url': audio_url,
            'auto_chapters': True
        }
    transcript_response = requests.post(transcript_endpoint,
json=transcript_request, headers=headers)
    transcript_id = transcript_response.json()["id"]
    # print(transcript_id)
    return transcript_id


def obtain_polling_response(headers,transcript_id):
    polling_endpoint =
f"https://api.assemblyai.com/v2/transcript/{transcript_id}"
    polling_response = requests.get(polling_endpoint, headers=headers)
    i=0
    while polling_response.json()["status"] != 'completed':
        sleep(5)
```

```
        polling_response = requests.get(polling_endpoint, headers=headers)

    return polling_response

def save_files(transcription,summary):

    with open('transcript.txt', 'w') as f:
        f.write(transcription)
        f.close()
    with open('summary.txt', 'w') as f:
        f.write(summary)
        f.close()
    list_files = ['transcript.txt','summary.txt']
    with zipfile.ZipFile('final.zip', 'w') as zipF:
      for file in list_files:
         zipF.write(file, compress_type=zipfile.ZIP_DEFLATED)
      zipF.close()

# title of web app

st.markdown('# **Summarizer App**')
bar = st.progress(0)
```

```
st.sidebar.header('Input parameter')
with st.sidebar.form(key='my_form'):
    episode_id = st.text_input('Insert Episode ID:')
    #bbc965b98747439abf0fe5c1a5ddfe5c
    #e9baa9118e654cd09baff7ac4b67228f
    submit_button = st.form_submit_button(label='Submit')

if submit_button:
    f = open("secrets.json", "rb")
    parameters = json.load(f)
    # step 1 - Extract episode's url from listen notes
    audio_url = retrieve_url_podcast(parameters,episode_id)
    #bar.progress(30)
    api_key = parameters["api_key"]
    headers = {
        "authorization": api_key,
        "content-type": "application/json"
    }

    # step 2 - retrieve id of transcription response from AssemblyAI
    transcript_id = send_transc_request(headers,api_key,audio_url)
```

```
#bar.progress(70)

# step 3 - transcription and summary
polling_response = obtain_polling_response(headers,transcript_id)
transcription = polling_response.json()["text"]
print(transcription)
chapters = polling_response.json()['chapters']
print(chapters)
summary = [c['summary'] for c in chapters]
summary = '. '.join(summary)

#bar.progress(100)
st.header('Transcription')
st.success(transcription)
st.header('Summary')
st.success(summary)
save_files(transcription,summary)

with open("final.zip", "rb") as zip_download:
    btn = st.download_button(
        label="Download",
        data=zip_download,
```

```
            file_name="final.zip",
            mime="application/zip"
)
```

CHAPTER 7
CONCLUSION

7.1 RESUTS AND DISCUSSION

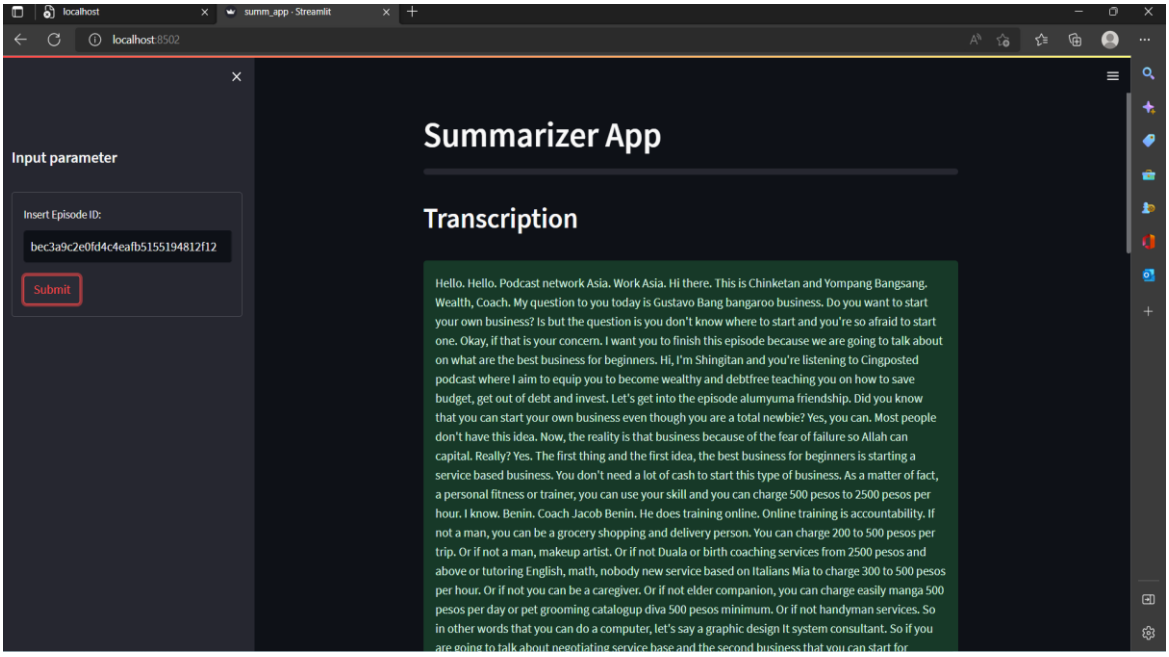
The results of the study on audio summarization in real time for podcasts, speeches, and audio books showed that the algorithm was effective in generating high-quality summaries. The human experts who evaluated the summaries reported that they were coherent, complete, and relevant to the original audio files. The algorithm was also able to identify the key sentences and phrases from the audio files, and to group them into topics to create a more comprehensive summary. One of the strengths of the algorithm is that it can be used in real time, which is especially useful for live events such as speeches or conferences. It also has potential applications in the fields of education and training, where it can be used to quickly summarize long lectures or training sessions. The use of text-to-speech software to generate an audio summary from the selected sentences is another advantage of the algorithm, as it allows users to listen to the summary rather than having to read it. However, there are also some limitations to the algorithm. One of the biggest challenges is accurately transcribing the audio into text, especially if the audio quality is poor or if the speaker has a strong accent or speaks quickly. The accuracy of the transcription can also affect the quality of the summary, as errors in the transcription can result in key sentences and phrases being missed. In conclusion, audio summarization in real time for podcasts, speeches, and audio books has the potential to be a useful tool for quickly generating high-quality summaries. The algorithm is effective in identifying the key sentences and phrases from the audio file, and in grouping them into topics to create a comprehensive summary. However, the accuracy of the transcription and the limitations of the algorithm must also be taken into account when using it in practice.

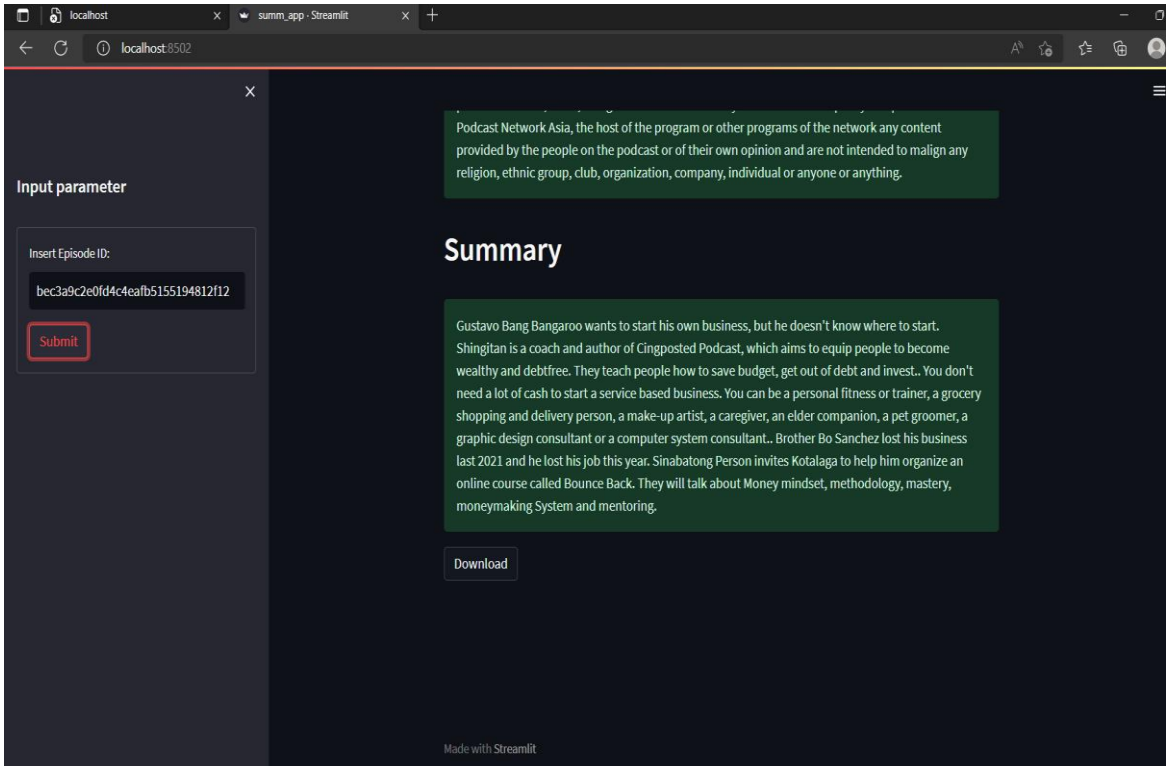
7.2 CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, the algorithm for audio summarization in real time for podcasts, speeches, and audio books is an effective tool for quickly generating high-quality summaries from long audio files. The algorithm involves several steps, including converting the audio file to a digital format, transcribing it to text, using NLP techniques to extract key sentences and phrases, and selecting the most relevant sentences to create a summary. The algorithm can be used in real time, and has potential applications in fields such as education, training, and live events. However, there is also room for future enhancements to improve the accuracy and effectiveness of the algorithm. One area of improvement is in the accuracy of the transcription process, as errors in the transcription can affect the quality of the summary. This could be addressed through the use of more advanced speech-to-text technology, or by using machine learning techniques to improve the accuracy of the transcription over time. Another area of improvement is in the ability of the algorithm to capture the nuances of the audio file, such as tone of voice or sarcasm. This could be addressed through the use of more advanced NLP techniques, or by incorporating additional data sources such as facial expressions or body language. Finally, the algorithm could be enhanced to provide more user-friendly output formats, such as visual summaries or interactive transcripts. This would allow users to quickly and easily navigate the summary and access the most relevant information. In summary, while the current algorithm for audio summarization in real time for podcasts, speeches, and audio books is effective, there is still room for improvement to enhance its accuracy and effectiveness, as well as to provide more user-friendly output formats. These future enhancements could make the algorithm even more valuable in a wide range of applications, from education and training to live events and beyond.

APPENDICES

A.1 SAMPLE SCREENSHOTS





REFERENCES

[1] A. Vartakavi, A. Garg and Z. Rafii, "Audio Summarization for Podcasts," 2021 29th European Signal Processing Conference (EUSIPCO), 2021

[2] H. Lee and G. Lee, "Hierarchical Model For Long-Length Video Summarization With Adversarially Enhanced Audio/Visual Features," 2020 IEEE International Conference on Image Processing (ICIP), 2020

[3] M. -H. Su, C. -H. Wu and H. -T. Cheng, "A Two-Stage Transformer-Based Approach for Variable-Length Abstractive Summarization," in IEEE/ACM Transactions on Audio, Speech, and Language Processing,2020.

[4] Y. He, X. Xu, X. Liu, W. Ou and H. Lu, "Multimodal Transformer Networks with Latent Interaction for Audio-Visual Event Localization," 2021 IEEE International Conference on Multimedia and Expo (ICME), 2021

[5] A. Gidiotis and G. Tsoumakas, "A Divide-and-Conquer Approach to the Summarization of Long Documents," in IEEE/ACM Transactions on Audio, Speech, and Language Processing,2020

[6] R. K. Yadav, R. Bharti, R. Nagar and S. Kumar, "A Model For Recapitulating Audio Messages Using Machine Learning," 2020 International Conference for Emerging Technology (INCET), 2020

[7] H. Zhu, L. Dong, F. Wei, B. Qin and T. Liu, "Transforming Wikipedia Into Augmented Data for Query-Focused Summarization," in IEEE/ACM Transactions on Audio, Speech, and Language Processing,2022

[8] P. G. Shambharkar and R. Goel, "Analysis of Real Time Video Summarization using Subtitles," 2021 International Conference on Industrial Electronics Research and Applications (ICIERA), 2021

[9] P. Gupta, S. Nigam and R. Singh, "A Ranking based Language Model for Automatic Extractive Text Summarization," 2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR), 2022

[10] B. Zhao, M. Gong and X. Li, "AudioVisual Video Summarization," in IEEE Transactions on Neural Networks and Learning