College of Computing and Informatics

Computer Science Department

University of Sharjah

Fall 2023/2024

LASER: Learning from Automatically Summarized and Segmented Educational Recordings

A project submitted
in partial fulfillment of the requirements for the degree of
Bachelor in Computer Science

by

Yaser Haitham Alesh (U20105776)

Osama Abdulghani (U20100427)

Meriem Aoudia (U20102795)

Omar Ibrahim Al Ali (U20102328)

Supervised by

Manar Abu Talib

ACKNOWLEDGEMENT

We extend our heartfelt gratitude to everyone who has played a pivotal role in the successful completion of our graduation project. The culmination of our efforts would not have been possible without the unwavering support and guidance from various individuals.

First and foremost, we express our deepest appreciation to Dr. Manar, our esteemed project supervisor. Dr. Manar's expertise, encouragement, and insightful feedback have been invaluable throughout the development of our project. The project benefited immensely from Dr. Manar's dedication to academic excellence and commitment to fostering our growth as students.

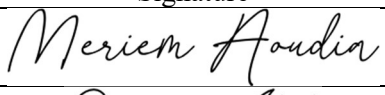
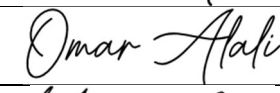
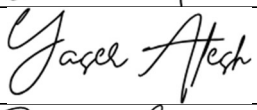
Next, we would like to acknowledge and express gratitude to our team members for their collaborative efforts. The successful integration of the four major components of our system into a seamless web application was made possible by the dedication and teamwork of each member.

Finally, a special note of appreciation goes out to our fellow students who served as testers for our system. Their participation played a crucial role in ensuring the reliability and user-friendliness of the final product.

UNDERTAKING

This is to declare that the project entitled "LASER: Learning from Automatically Summarized and Segmented Educational Recordings" is an original work done by undersigned, in partial fulfillment of the requirements for the degree "Bachelor's in Computer Science" at the Computer Science Department, College of Computing and Informatics, University of Sharjah, UAE.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

ID	Name	Signature
U20102795	Meriem Aoudia	
U20102328	Omar Ibrahim Mahmoud Alali	
U20100427	Yaser Haitham Alesh	
U20100427	Osama Mohamad Abdulghani	

Date: December 4th ,2023

ABSTRACT

In today's age of abundant online learning materials, LASER stands out as an innovative solution poised to transform the educational landscape. LASER, an acronym for Learning from Automatically Summarized and Segmented Educational Recordings, is a user-friendly web application. Its primary goal is to simplify the complexities often faced when engaging with recorded lecture videos.

This platform leverages state-of-the-art artificial intelligence technologies to offer a range of features. These include video summarization, segmentation, transcription, and question-answering capabilities. LASER is meticulously crafted to address the needs of educators, students, and lifelong learners alike.

By harnessing the power of AI, LASER seeks to enhance the learning experience by making educational content more accessible and digestible. The application is designed to overcome the challenges associated with navigating and comprehending lengthy lecture videos. Its intuitive interface ensures ease of use for individuals at all levels of technical proficiency.

LASER's impact extends beyond mere convenience. It aspires to revolutionize the traditional methods of consuming educational content. Through its advanced features, LASER aims to streamline the learning process, making knowledge acquisition more efficient and enjoyable. In essence, LASER is not just a tool; it represents a significant step forward in the evolution of online education.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: Introduction

Welcome to the world of LASER - Learning from Automatically Summarized and Segmented Educational Recordings. This innovative project brings together education and technology, aiming to revolutionize the way we interact with educational content. In this chapter, we delve into the background of the project, the motivation that propels us, the problems we seek to address, and the expected outcomes that await us on this exciting journey.

1.1 Overview

LASER, or Learning from Automatically Summarized and Segmented Educational Recordings, is a web application designed to streamline the process of learning from recorded lecture videos. It integrates AI to help educators, students, and everyone to digest learning material easily and fast. The application focuses on four major components: transcribing, summarizing, segmenting, and question answering.

The need for such a tool is evident in the recent surge of online learning. More than 20 million new learners registered for online learning in 2021 [10]. Figure 1 shows the number of new registered learners for online courses.

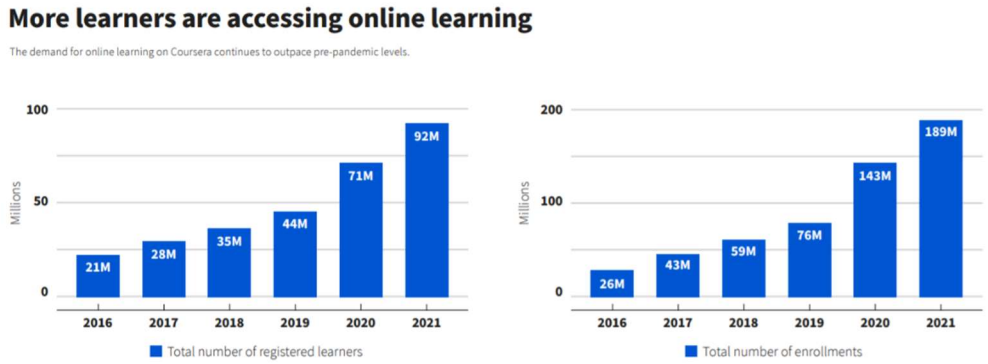


Figure 1. Over 20 Million Individuals Enrolled in Courses During the Year 2021 [16]

Furthermore, educational videos are the fourth most commonly consumed content on the internet [11][12]. However, the process of learning from these videos can be time-consuming and requires manual effort. This is where LASER comes in.

LASER aims to automate the process of learning from these videos. It focuses on the

automatic segmentation and summarization of educational and lecture videos. These tasks are challenging due to the complex nature of video content. Educational and lecture videos differ significantly from other types of videos, such as news and sports, which are commonly addressed in the video summarization field [13][14]. They are typically heavy with textual and graphical elements, making it necessary to develop new techniques to tackle this problem [15].

The development of LASER is backed by extensive research. A comprehensive study of automatic video summarization techniques has been conducted, and the application is being developed based on the findings [13]. The goal is to enhance the scope and performance of automatic video summarization systems, and by doing so, make the process of learning from recorded lecture videos more efficient and effective.

1.2 Project Motivation

The motivation behind the development of LASER (Learning from Automatically Summarized and Segmented Educational Recordings) stems from the recognition of the evolving landscape of education and the increasing need for efficient learning tools. In today's fast-paced world, where information is rapidly produced and consumed, one of the ways to obtain and remember information efficiently is through summarization [17]. Studies show that students who are successful in reading comprehension are also successful in summarization [17]. Furthermore, transcribing educational research materials promotes accessibility and enhances the overall quality of the research itself [18].

The importance of segmenting in learning is also well recognized. Segmenting refers to the process of breaking down complex lessons into smaller parts, which are then presented one at a time [19]. This process helps learners manage the complexity of the content/tasks they are presented with, so as not to exceed the threshold of their cognitive system [19].

Furthermore, questioning is an important activity in teaching. It can be used to test the knowledge of the past, stimulate student thinking, and is a sign of understanding, not ignorance [20] [21]. Table 1 shows the strengths and weaknesses of questions in learning.

Table 1.Comparative Analysis: Strengths and Weaknesses of Questions in Learning [21]

Relative Strengths of Questions	Relative Weaknesses of Questions
Good questions can reveal subtle shades of understanding – what this student knows about this topic in this context	Questions depend on language, which means literacy, jargon, confusing syntax, academic diction, and more can all obscure the learning process
Questions promote inquiry and learning how to learn over proving what you know	Questions can imply answers, which imply stopping points and 'finishing' over inquiry and wisdom (See questions that promote inquiry-based learning.)
Questions fit in well with the modern 'Google' mindset	Accuracy of answers can be overvalued, which makes the confidence of the answerer impact the quality of the response significantly
Used well, questions can promote personalized learning as teachers can change questions on the fly to meet student needs	"Bad questions" are easy to write and deeply confusing, which can accumulate to harm a student's sense of self-efficacy, as well as their own tendency to ask them on their own

The LASER project is important because it integrates these four major components - transcribing, summarization, segmenting, and question answering - into a single web application. This integration is the new idea proposed by this project. By leveraging AI, LASER aims to help educators, students, and everyone to digest learning material easily and quickly. It is designed to enhance the learning experience, improve comprehension, and promote efficient learning. This project is not just about creating a tool, but about contributing to the evolution of learning in the 21st century.

1.3 Problem Statement

The LASER project addresses several key issues in the current educational landscape.

1- Firstly, the traditional model of education often involves a linear and one-size-fits-all approach, which may not cater to the diverse learning needs and styles of students [22][23]. This is where the component of AI-driven summarization in LASER comes in. Summarization has been shown to be an effective tool for improving reading comprehension [22], and by integrating this into a web application, we aim to make learning more personalized and efficient.

2- Secondly, accessibility is a significant issue in education. Transcribing educational

materials can enhance accessibility and improve the quality of research [22]. By incorporating a transcription component, LASER aims to make educational content more accessible to a wider audience.

3- Thirdly, the complexity of educational content can often be overwhelming for learners. The process of segmenting, or breaking down complex lessons into smaller parts, can help manage this complexity [22]. LASER integrates this process into its system, aiming to make learning more manageable and less overwhelming for users.

4- Lastly, the importance of questioning in teaching is well recognized [22]. However, formulating effective questions can be challenging. LASER includes a question-answering component, leveraging AI to generate meaningful questions that stimulate thinking and deepen understanding.

In addition to addressing these issues, LASER also aims to improve upon existing conditions in the educational landscape. In summary, the LASER project seeks to address key issues in education and improve upon existing conditions by integrating AI into a comprehensive web application for learning.

1.4 Project Aim and Objectives

The primary aim of the LASER project is to revolutionize the way educational content is consumed by integrating Artificial Intelligence (AI) into a web application. This project seeks to make learning more accessible, efficient, and engaging by transcribing, summarizing, segmenting, and answering questions from YouTube or MP4 video files [24][25][26].

To achieve this goal, the project will focus on the following objectives:

1- Transcribing: The application will convert spoken language in videos into written text. This feature will aid in understanding the content, especially for individuals who might have hearing impairments or those who prefer reading over listening.

2- Summarization: Leveraging AI, the application will provide concise summaries of the video content. This will help users to quickly grasp the key points without having to watch the entire video. This is particularly beneficial given that 74% of trainers use video learning as part of their training delivery [24]. Figure 2 demonstrates the utilization of corporate learning technology categorized by type.

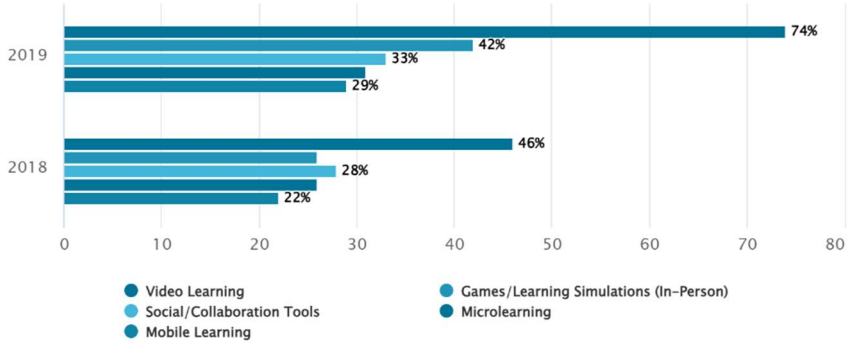


Figure 2. Usage of Corporate Learning Technology by Type [24]

3- Segmenting: The application will break down the video content into manageable segments or chapters. This feature aligns with the trend towards bite-sized learning, which has been shown to increase learner engagement and participation [26][27].

4- Question Answering: The application will include a feature to answer questions based on the video content. This interactive element can enhance the learning experience by promoting active engagement with the material.

By integrating these components, the LASER project aims to address the growing demand for video-based learning, which is preferred by 83% of people over text or audio-based instructional content [25]. Furthermore, it aligns with the increasing adoption of AI in education, which has the potential to address some of the biggest challenges in education today [29][26][30]. Ultimately, the LASER project strives to make learning more accessible and engaging, catering to the diverse needs of educators, students, and lifelong learners alike.

1.5 Project Scope

The LASER initiative, which stands for Learning from Automatically Summarized and Segmented Educational Recordings, represents a comprehensive web application poised to transform the paradigm of engaging with educational content. This collaborative effort involves four dedicated computer science students, each contributing their unique expertise to develop a tool that incorporates AI to facilitate a more efficient and effective consumption of learning material from YouTube or MP4 video files.

The project is characterized by its four primary components: transcribing, summarization, segmenting, and question answering. The transcribing element plays a pivotal role in converting spoken language within video files into written text, laying the groundwork for subsequent processes. Following this, the summarization component condenses the transcribed text into a concise format, preserving essential information for quick comprehension. The segmenting component further enhances user experience by breaking down the summarized text into manageable segments, facilitating improved understanding and retention of information. Lastly, the question-answering component introduces an interactive feature that responds to user queries, augmenting the overall learning experience.

To ensure the project's integrity, each team member was assigned a specific component, guaranteeing focused attention on every aspect. Active verification and approval processes have been instituted to uphold the project's quality and functionality. Comprehensive testing, both individually and as part of the integrated system, occurred through regular team meetings and code reviews. The project reached its conclusion when all components seamlessly functioned together.

1.6 Project Software and Hardware Requirements

On the software front, the application was built on a modern web development framework, integrating React for the frontend and Flask for the server-side environment. The AI components of the project were developed using Python. The application was designed to be compatible with contemporary web browsers, specifically Google Chrome and Microsoft Edge.

In terms of hardware, the server hosting the application needed to be equipped with a high-speed processor and sufficient memory to handle the computational demands of the AI components. Notably, a powerful GPU was used for the fine-tuning and training of the different components, given the intensive computational requirements of these processes. Additionally, a stable and high-speed internet connection was essential for seamless video processing and user interaction.

1.7 Project Limitations

While the LASER project was designed to be a comprehensive tool for enhancing the learning experience, it did have certain limitations. The version of the application that was developed supported only YouTube and MP4 video files. The accuracy of the transcribing, summarization, and question-answering components was dependent on the clarity of the spoken language in the video content.

1.8 Project Expected Output

The expected output of the LASER project was a fully functional web application that provides an enhanced learning experience. Users can upload YouTube or MP4 video files and receive transcribed, summarized, and segmented text. Additionally, the application provided an interactive question-answering feature that could respond to user queries related to the video content.

1.9 Project Schedule

Timeline	Phase Title	Tasks
December 2022 - January 2023	**Project Initiation and Planning**	1. Gather project team.
		2. Establish communication channels.
		3. Break down tasks based on junior report requirements.
		4. Assign responsibilities to team members.
February 2023 – March 2023	**Research and Data Collection**	1. Dive deeper into literature.
		2. Identify and compile relevant datasets.
		3. Develop a plan for collecting data.
		4. Create a new dataset for summarization, segmentation, and ASR.
April 2023 – May 2023	**Model Development and Experimentation**	1. Experiment with the Whisper model.
		2. Fine-tune the Whisper model using a special library.
		3. Evaluate pre-trained models for summarization.
		4. Fine-tune selected models using HuggingFace and PyTorch libraries.
September 2023 - November 2023	**Web Application Development**	1. Finalize technical requirements.
		2. Design the architecture for the LASER platform.
		3. Develop video summarization and segmentation features.
		4. Implement transcription and question-answering capabilities.
December 2023	**Project Conclusion and Reporting**	1. Conduct thorough testing of LASER features.
		2. Address any bugs or issues.
		3. Summarize the achievements and features of LASER.
		4. Reflect on the overall impact and significance.
		5. Prepare a presentation for stakeholders.
		6. Document the entire project for future reference.

Ongoing Tasks Throughout the Project:

1- Regular team meetings and bi-weekly meetings with the supervisor .

2- Continuous refinement of models based on experimentation results and debugging.

3- Regular updates to the project timeline based on progress and challenges encountered.

1.10 Project, product, and schedule risks

One of the major risks associated with the LASER project was the potential for the project to take longer than scheduled. This could have been due to unforeseen complexities in the development of the AI components, difficulties in integrating the components, or issues encountered during the testing phase. To mitigate this risk, the team conducted regular meetings to monitor progress and address any issues promptly. Additionally, emergency time was factored into the schedule to accommodate potential delays.

CHAPTER 2: Related Existing System

2.1 Introduction

This chapter provides an overview of existing systems related to lecture summarization and explores their features, strengths, and weaknesses. The goal is to establish a foundation for understanding the landscape of available solutions and to position our project, LASER, as a superior option for students, educators, and anyone seeking advanced features such as summarization, segmentation, transcription, and question-answering for lengthy YouTube videos and video MP4 files.

2.2 Existing Systems

2.2.1 Meetings and Lectures

Fireflies.ai: Fireflies stands out as an AI notetaker primarily tailored for meeting environments. Its strengths lie in generating highly accurate transcripts and summaries promptly, aiding in the extraction of actionable insights. It excels in collaboration tools, extensive integration capabilities with various business apps, and support for multiple languages. However, it heavily relies on an internet connection, lacks translation options, and comes with pricing plans based on usage tiers.

Vowel AI: Vowel AI specializes in live transcription and summary generation during online meetings. Offering features like live captioning, action item identification, and speaker identification, it delivers real-time insights. Yet, its limited integration and occasional background noise recognition issues pose challenges.

2.2.2 YouTube Video Summarization

Eightify and Summarize.tech: These tools focus on condensing lengthy YouTube videos into concise summaries. Eightify excels in reducing videos into eight key points quickly, albeit with limited customization and exclusivity to YouTube content. Summarize.tech, while supporting longer videos, lacks language diversity and download options.

2.2.3 Text Summarization

Jasper.ai, Quillbot, and Frase.io: These systems cater to text content, providing options to summarize articles or generate reports. Jasper and Quillbot offer multi-format summaries with varying levels of language support and pricing structures. Frase.io, though limited in functionalities, provides quick and free article summarization.

2.2.4 Web Page Summarization

Smodin and Website Summary AI: These tools aim to summarize web pages into condensed versions. Smodin offers customizable summaries for diverse content but may lack variety in phrasing. Website Summary AI, though free, imposes limitations on the number of summaries per hour and content-specific effectiveness.

2.3 Overall Problems of Existing Systems

To understand the landscape of existing summarization tools, we conducted a comprehensive review of prominent solutions, namely Fireflies.ai, Vowel AI, Eightify, Summarize.tech, Jasper.ai, Quillbot, Frase.io, Smodin, and Website Summary AI. A comparative analysis of their pros and cons is summarized in the table below:

Table 2. Comparison of Similar Systems

References	Tool	Pros	Cons
[1]	Fireflies.ai	- 90% accurate transcript and summary - Advanced AI filters and Smart Search - Multi-language transcription - Integrations with various apps	- Requires a stable internet connection - No in-built translation options
[2]	Vowel AI	- Live captioning - Multi-language transcription - Timestamps and speaker identification	- Limited integrations - Background noise pickup
[3]	Eightify	- Turbo GPT-powered YouTube video summary. - Chrome extension for one-click summaries	- Limited to YouTube videos - No customization of summary length/format
[4]	Summarize.tech	- Fast and accurate summarization - No daily limits. - Supports videos of any length	- Supports only English language - No transcript download

[5]	Jasper.ai	- Supports 30+ languages - Faster content creation - Voice commands and tone setting	- Expensive for startups - Challenges with overly technical topics
[6]	Quillbot	- Multiple summary formats - In-built plagiarism detection and grammar checker	- Tone inconsistency - Limited language support
[7]	Frase.io	- Free summary generation tool - SEO-optimized content briefs in seconds	- Limited integrations - No multi-language support
[8]	Smodin	- Customizable summary length and content classes - Translator and plagiarism checker	- Generates summaries in one format - Repetitive content
[9]	Website Summary AI	- Generates summary in one click - Natural language prompts for customization	- Limits on summaries per hour - Results vary based on website content

2.4 Overall Solution Approach

LASER, our cutting-edge educational solution, is designed to cater to the diverse needs of students, educators, and knowledge seekers. It's a revolutionary platform that transforms content processing to enhance learning and productivity significantly. The platform offers a myriad of features, each tailored to optimize the educational experience:

- **Summarization with Context Preservation:** Leveraging state-of-the-art algorithms, LASER intelligently condenses lengthy videos and text documents into succinct yet comprehensive summaries. This process doesn't just capture key points but also maintains the contextual essence, ensuring a good understanding of the content.
- **Segmentation for Enhanced Comprehension:** LASER breaks down complex materials into manageable sections, empowering users to navigate through intricate content with ease. By offering segmented content, users can grasp challenging concepts more effectively, enhancing overall comprehension.
- **Transcription in Multiple Formats for Accessibility:** Accessibility is prioritized with LASER's robust transcription feature. It transcribes the audio content from various sources like YouTube or an mp4 video file uploaded by the user, enabling users to access transcriptions in formats that suit their needs.
- **Advanced Question Answering System:** The platform hosts an interactive

question-answering system that fosters deep engagement. Users can interact with the content by asking queries and receiving detailed responses, thereby solidifying their understanding of the material.

- **Personalized Profile Page with Downloadable Transcripts:** LASER offers a centralized profile page where all processed videos and transcriptions are stored. Users can easily access and download transcripts from this page, ensuring seamless retrieval and utilization of processed content.

LASER stands apart by merging the strengths of existing solutions into a versatile, user-centric platform. Our overarching goal is to redefine the learning and productivity experience for our users, offering a comprehensive suite of tools to optimize knowledge acquisition and retention.

In the upcoming chapter, we will delve into the realm of datasets, a pivotal aspect in shaping LASER's capabilities. Datasets play a crucial role in fueling our innovative educational platform. This section will provide a detailed account of the datasets utilized, shedding light on their significance in tasks such as summarization, title generation, and segmentation.

CHAPTER 3: Datasets

3.1 Datasets

In this section of the report, we will discuss what datasets we have and what we have used them for. First, we have researched what datasets there are available and ready to on the internet regarding the summarization, title generation, and segmentation task. Unfortunately, we haven’t found any publicly available datasets that we would benefit and our process much easier and faster to accomplish. So, we decided to go with the web scraping techniques, where we can extract or scrape data ethically from the internet. We have found that the AK lectures website [56] and MIT OpenCourseWare YouTube channel [57] benefit us a lot in order to train some AI models that accomplishes our desired features. The tools of web scraping we used are selenium and beautiful soup frameworks. We already have discussed in detail what we did to create the datasets in the Junior Final Report. Here, we will revise and explain more what we have used them for.

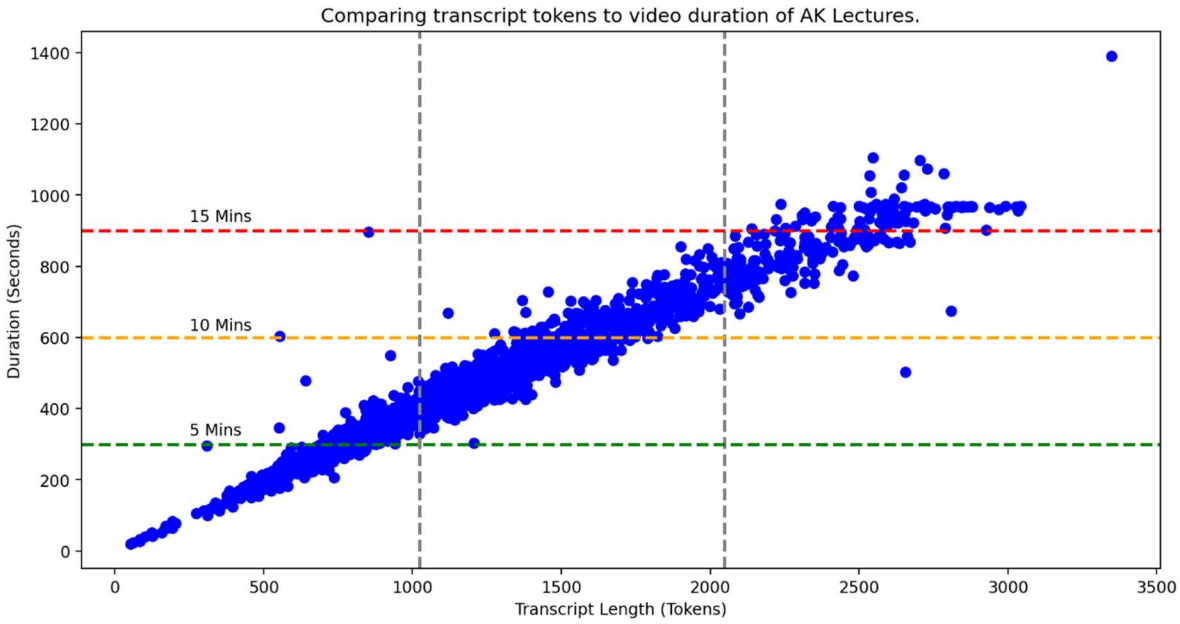


Figure 3. Comparing Transcript Tokens to Video Duration of AK Lectures

A plot of the video duration vs. transcript length (in tokens) of all videos in the AK Lectures dataset. We notice that videos less than 10 minutes in length always have less than 2000 tokens. Figure 4 shows a comparison between the transcript length of all videos in AK Lectures vs. the compressed transcript after extractive summarization has been performed. Compares two techniques: TextRank and LSA, which have shown the best result after the base configuration. And Figure 5 shows a comparison between the transcript length (in tokens) vs. the abstractive summary length of AK Lecture videos.

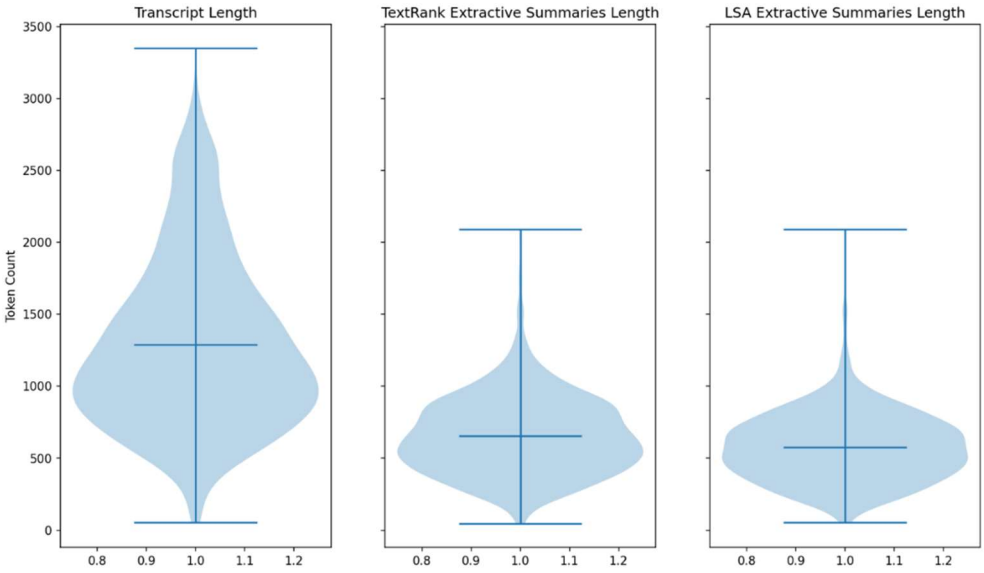


Figure 4. Comparative Analysis of Transcript Length

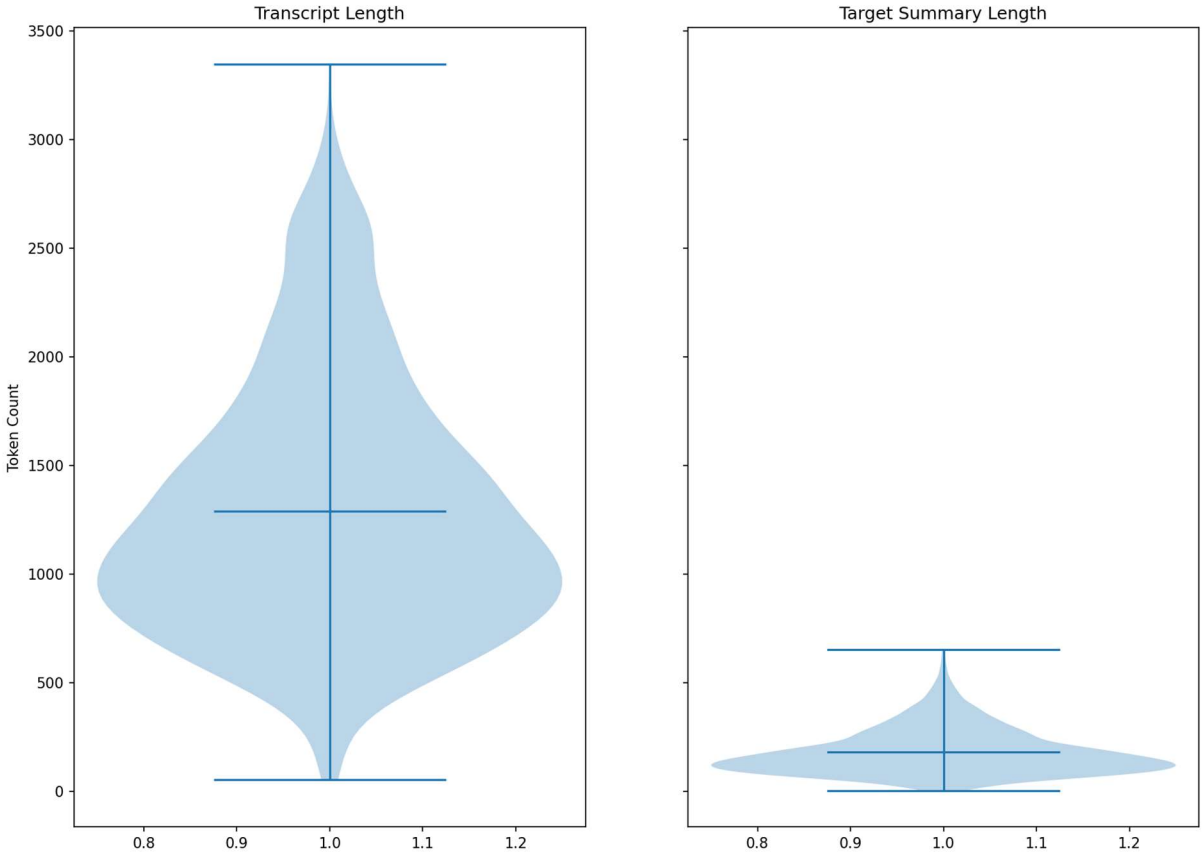


Figure 5. Token-based Analysis

3.2 MIT chapters

We have extended the AK Lectures dataset by creating the MIT Chapters dataset. This was motivated by different reasons:

1- The need for quality transcription and audio files pairs data for the evaluation and fine tuning of ASR and punctuation restoration methods.

2- To generate summaries for each segment in order to train summarization and title generation model.

3- To train our High-Quality segmentation model.

4- To evaluate our Segmentation techniques such as the Segmentation algorithm and Segmentation model.

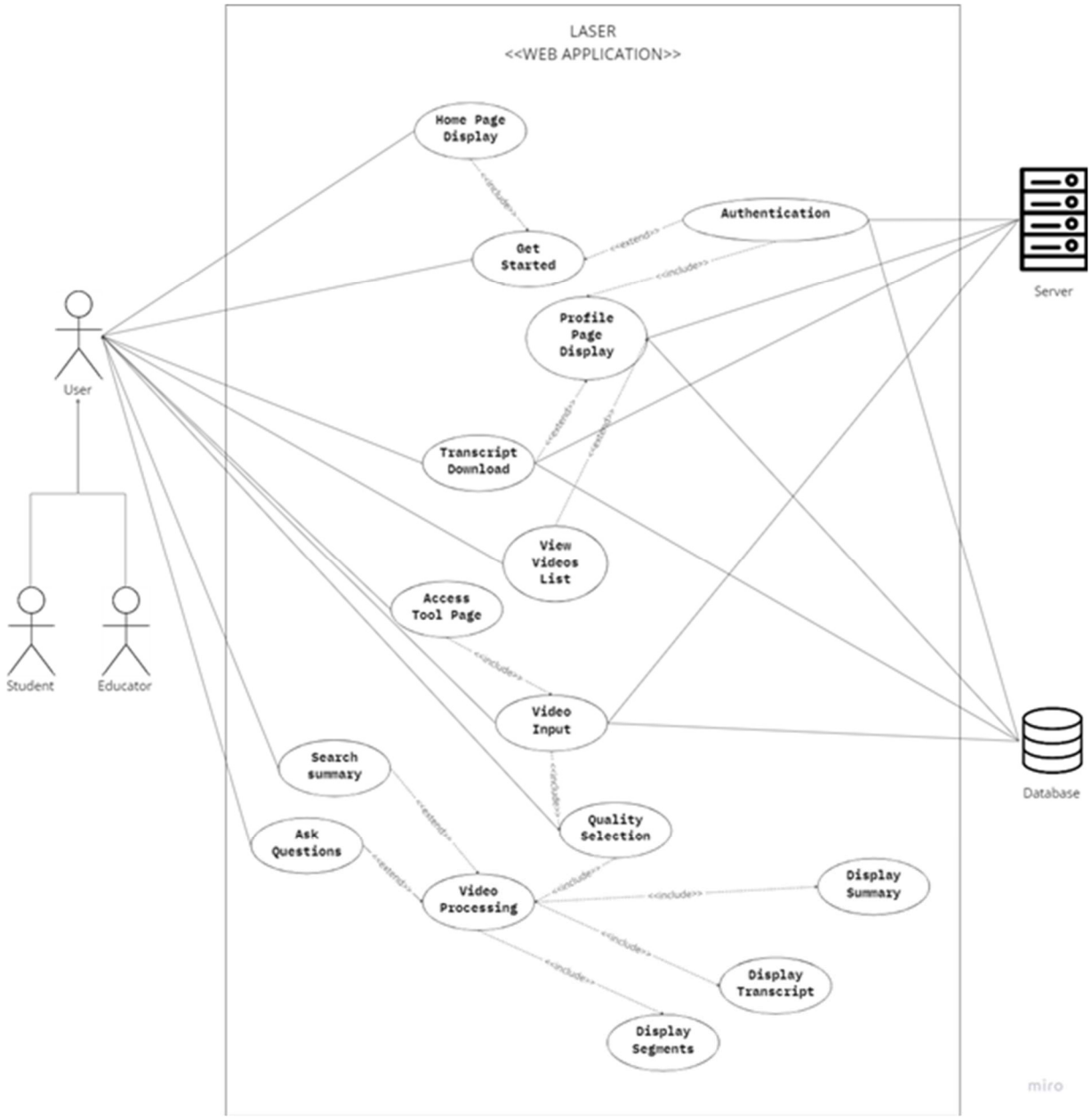
The MIT Chapters dataset consists of chapters of videos from MIT courses. The chapters have either been annotated by a human on YouTube, or automatically generated by YouTube algorithms. Each chapter has a title, a duration, a time slice, and a corresponding .wav file that has the audio from the original video corresponding to this chapter.

Note: When we started to evaluate the segmentation methods using this dataset. We deleted lectures that had small duration of time and AI generated chapter segments. We did this because we wanted the human annotated segments since its more accurate to use to evaluate.

CHAPTER 4: Architecture and Design

4.1 Software design

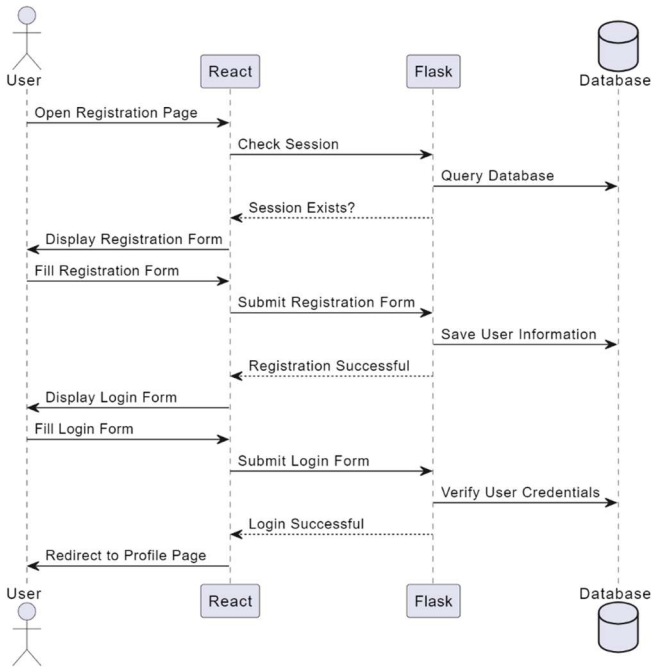
4.1.1 Use Case Diagram



The use case diagram depicts the functional interactions within LASER, the AI tool web application designed for students and educators. Three main actors shape the system: the User, who initiates the process, the Database, which stores user data and videos, and the Server, a Flask backend responsible for authentication, video processing, and data retrieval. The journey begins with the User accessing the Home Page and clicking "Get Started." Authentication checks occur on the Server, determining whether the User is directed to the Profile Page or prompted to register/login. Once in the Tool Page, the User engages in various tasks, including selecting the quality of segmentation, providing video input (YouTube link or upload), and triggering the Server to process the video. The Server generates a transcript, summary, segmentation, and question answering, presenting these outputs on the Tool Page. Users can seamlessly navigate between the Profile Page and the Tool Page to review processed videos. Additionally, the functionality extends to allow Users to download transcripts in PDF format from the Profile Page. This use case diagram captures the comprehensive flow of interactions, emphasizing the dynamic relationship between the User, Server, and Database in delivering an integrated educational experience.

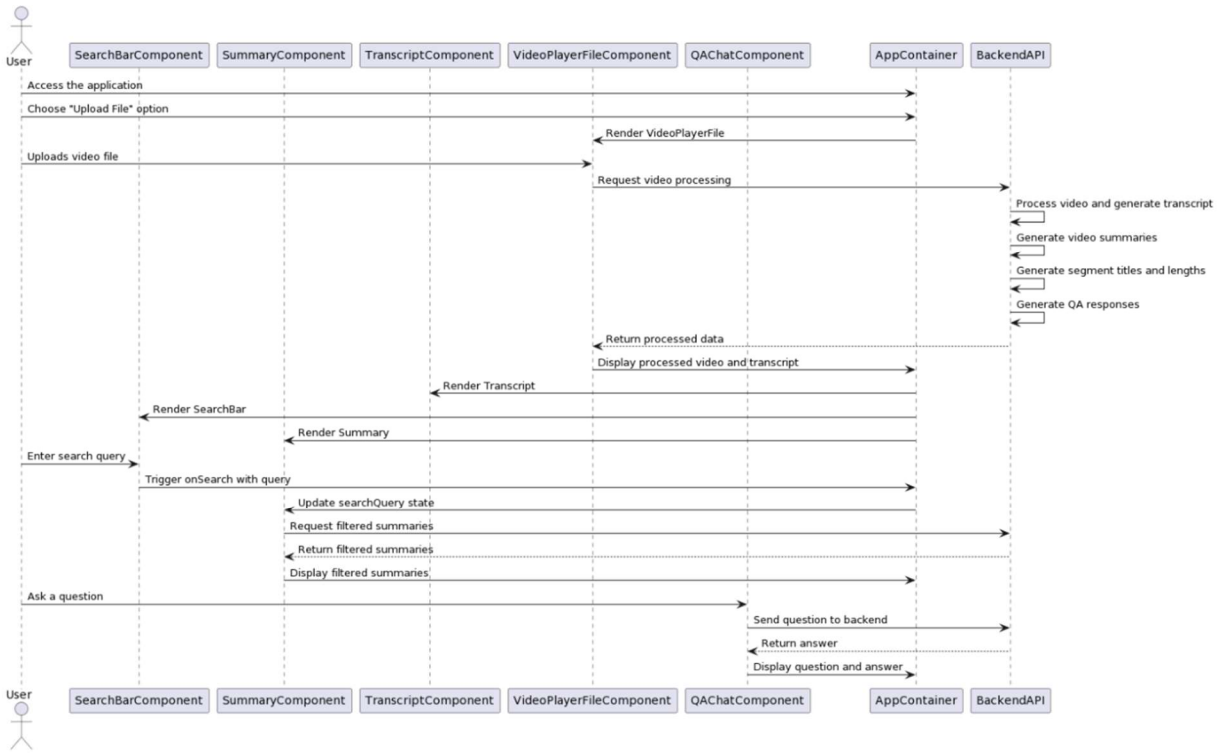
4.1.2 UML sequence/communication diagram

1- <u>Registration/Login Sequence Diagram</u>



In the LASER web application, the above sequence diagram illustrates the user registration and login process. Upon initiating the registration, the User interacts with the Frontend, triggering a session check with the Backend. The Backend queries the Database to determine the existence of a session, communicating the result back to the Frontend. If the session is absent, the Frontend prompts the User to complete the registration form, and upon submission, the Backend persists the user information in the Database, signaling a successful registration to the Frontend. Subsequently, the User is presented with the login form, leading to a similar process of form submission, credential verification by the Backend against the Database, and ultimately a successful login acknowledgment to the Frontend. The user is then redirected to their profile page.

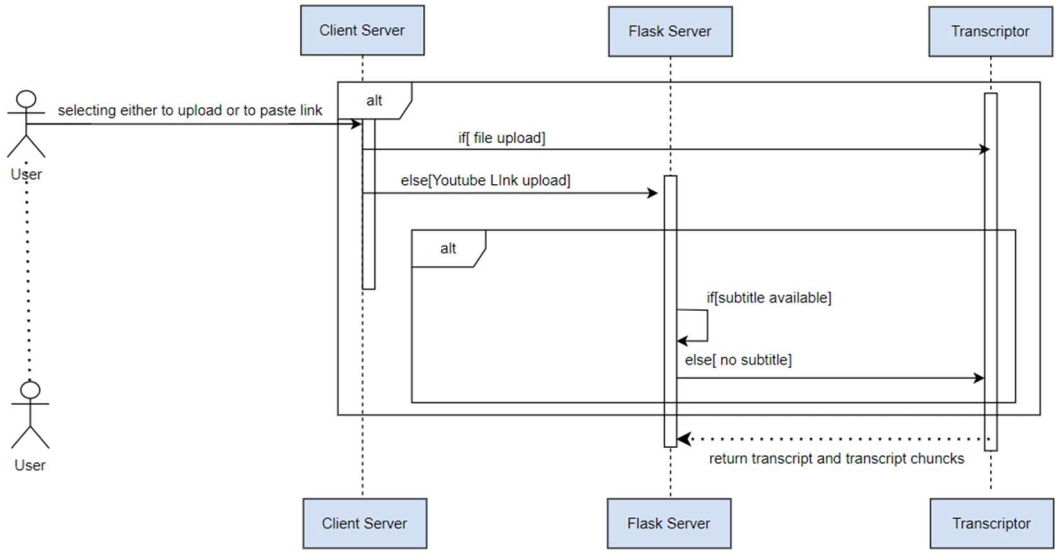
2- Application Workflow Sequence Diagram



This sequence diagram encapsulates the procedural flow of the application tool. Initiated by the user, the application journey commences with accessing and opting to upload a file. The AppContainer orchestrates the rendering of various components, including the VideoPlayerFileComponent, where the User uploads a video file. Subsequently, the VideoPlayerFileComponent communicates with the BackendAPI,

triggering video processing, transcript generation, and the creation of video summaries, segment titles, and QA responses. Upon completion, the BackendAPI returns the processed data to the VideoPlayerFileComponent, which, in turn, displays the enriched content in the AppContainer. The interface also encompasses a SearchBarComponent and SummaryComponent, with the former enabling users to enter search queries and the latter presenting filtered summaries. The AppContainer efficiently manages the synchronization of these components, updating the search query state and dynamically rendering pertinent information. Lastly, the QAChatComponent facilitates user inquiries, interacting seamlessly with the Backend API to retrieve and display responses within the AppContainer. This detailed sequence diagram provides a comprehensive overview of the application's functionalities and interactions.

3- Transcript Generation Sequence Diagram:

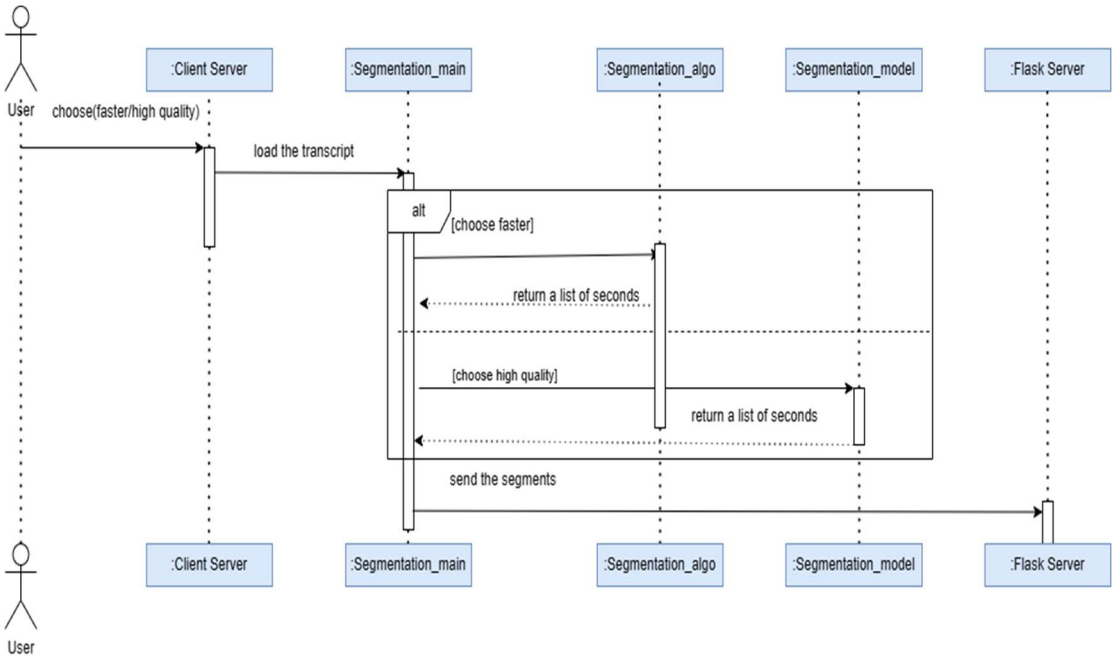


The diagram presented above illustrates the user journey upon entering the "Tools Page." Users are prompted to make a selection between two options: uploading a .mp4 file or providing a YouTube link. If the user chooses to upload a file, the content is directed to the Transcriptor class, where our Whisper model transcribes the content. Alternatively, if the user opts for a YouTube link, the information is transferred to the Flask Server. The server checks whether the YouTube video already contains subtitles, a favorable scenario as it allows us to bypass the transcription step, which is time-consuming. If subtitles are

present, their contents are returned to the Flask Server for further processing.

In the absence of subtitles for the provided YouTube link, the data is rerouted to the Transcriptor class. Here, the content undergoes transcription, and the results are subsequently sent from the Transcriptor to the Flask Server for subsequent operations. This systematic process ensures the efficient handling of user inputs, optimizing the workflow for both uploaded files and YouTube links within the LASER project's operational context.

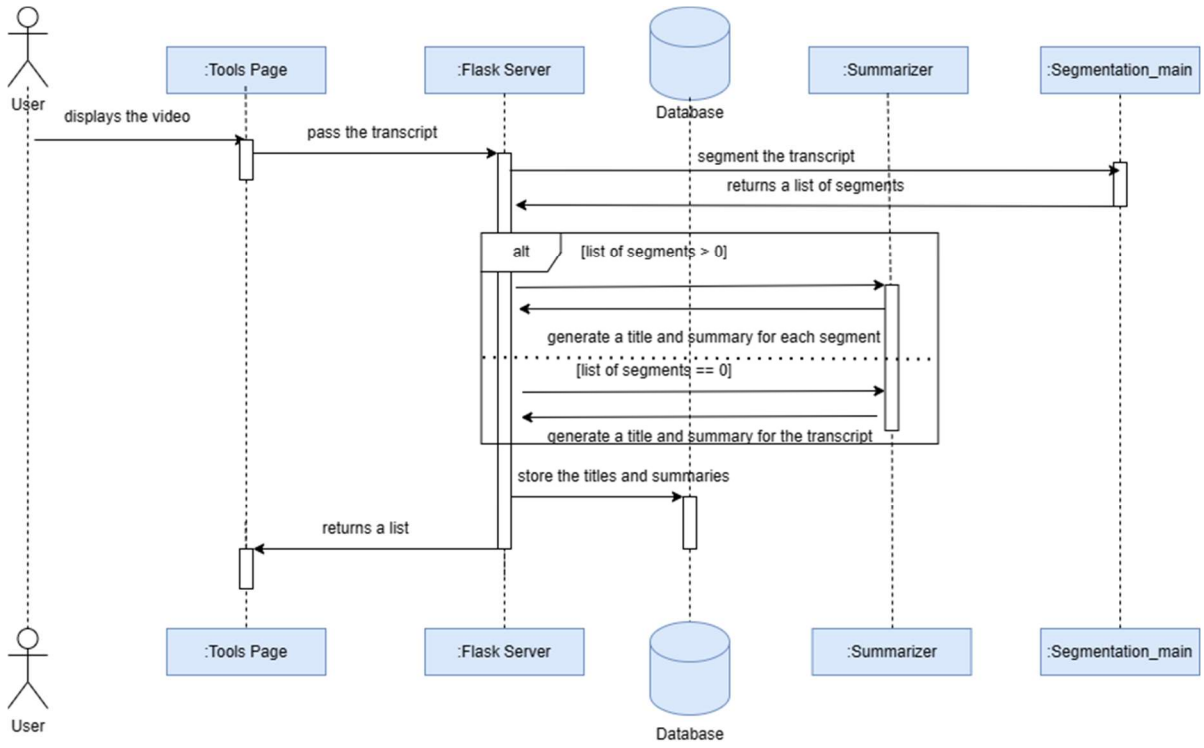
4- Segmentation Sequence Diagram:



The diagram shown above explains how the segmentation part of our project explains sequentially. The user first must choose one of the following two options “Faster” or “High quality”. If the user chooses the “Faster” option, that means they called the Segmentation algorithm class. If the user chooses the “High Quality” option, that means they called the Segmentation model class which is based on AI. The difference between those two models is that the Segmentation algorithm is just an algorithm “not based on AI” and it generates segments faster than the other model and the Segmentation model is an AI based model that is trained on the MIT dataset it generates more accurate segments than the Segmentation algorithm. After that, the user has also two options to choose from either to paste their YouTube link or upload one of their files, then the transcript model automatically starts to generate the transcript based on the speech. Once the transcript is

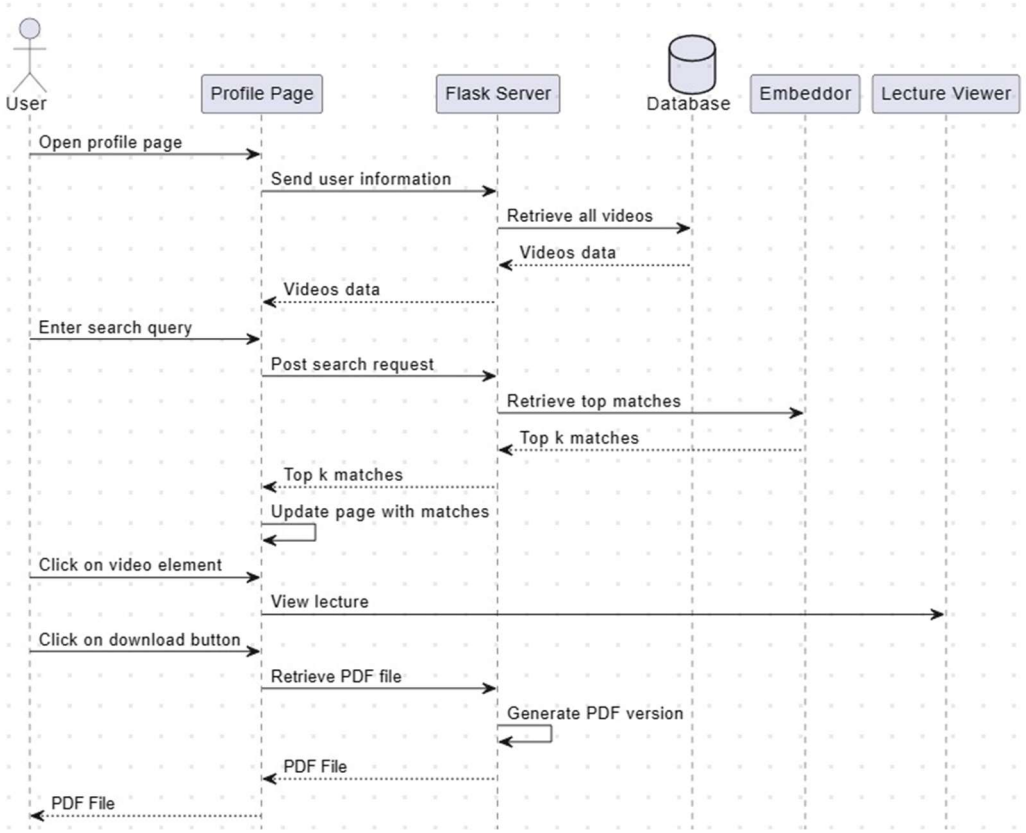
done, the transcript loads into the main segmentation class, we refer to it as "Segmentation_main". Then based on the option the user chose at the beginning which was either "Faster" or "High Quality" the segments get generated based on one of those options. At the end, we receive the segments as an array them we send it to the "Flask server" to continue its work.

5- Summarization and title generation Sequence Diagram:



After the transcript gets extracted and displayed on the "Tools" page, the "Flask server" requests from the "Segmentation_main" participant to segment the transcript. It segments the transcript using the 2 algorithms discussed in the previous diagram. Then, it returns a list of segments "filled with seconds". Once the "Flask server" receives the list, it checks whether the list is empty or not. If it's not empty, then the "Flask server" requests from the "Summarizer" participant to generate a title and summary for each segment. If it is empty, then the "Flask server" also requests from the "Summarizer" participant to generate only a summary for the whole transcript since it's not segmented. At the end, the "Flask server" returns a list of titles and summaries to be displayed on the "Tools" page.

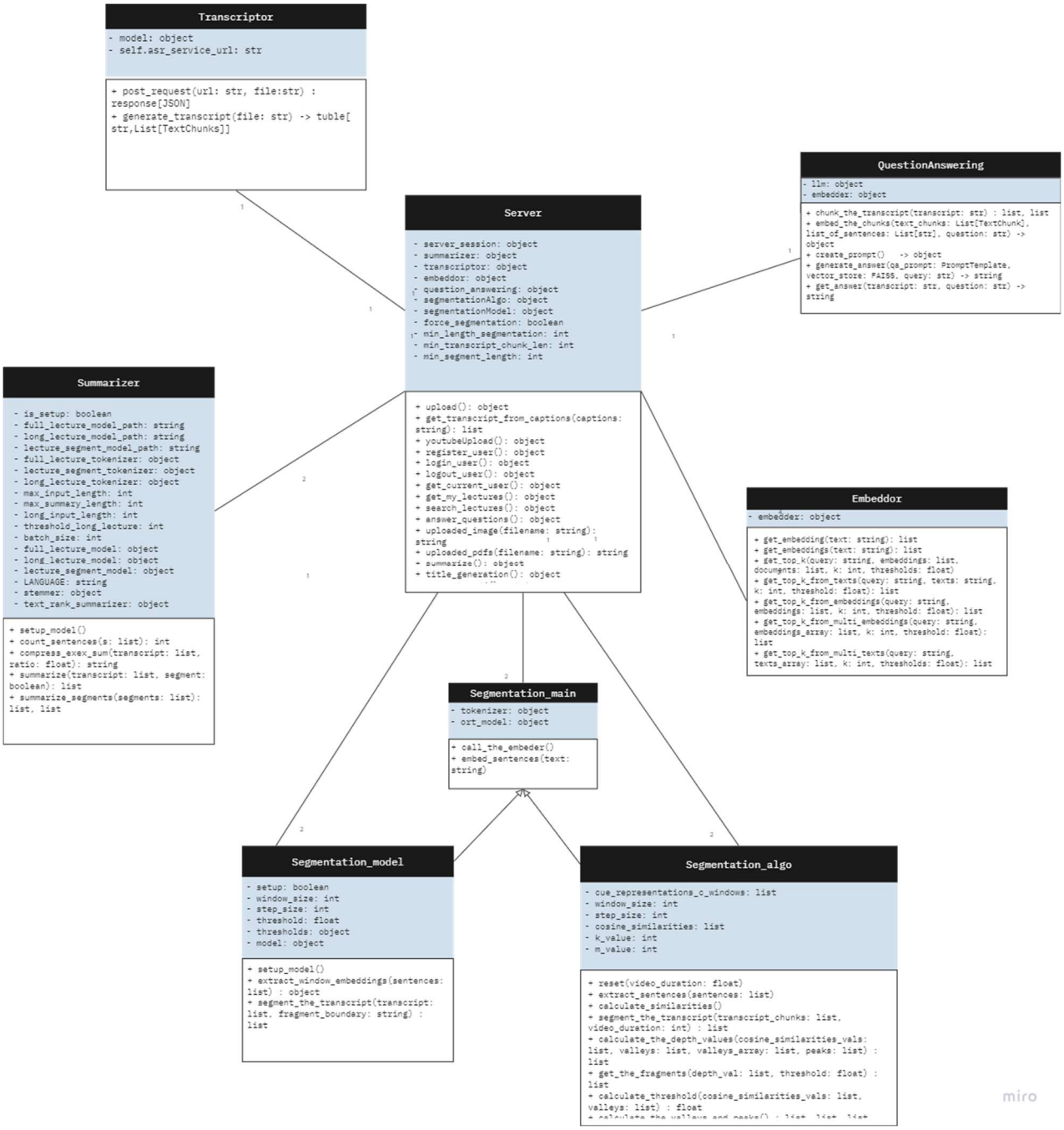
6- Searching Sequence Diagram:



We created a profile page for each user that had signed up and used the application multiple times, for each educational video they used in the application will be stored in the database and displayed in their own profile page. The user can go to his profile page and search for any video they previously used in the application. The process of searching goes as the following: the user types his/her search this search will be passed to the "Embeddor" through the "Flask server" to check if this search matches any video stored in the "Database". Then the "Embeddor" will return a list of top matches to the search then it updates the profile page based on the matches. When the matches get displayed, the user then can click on a video element to view the video or can click on the download button to download a pdf file that has the summaries he got when he used the video in the application before.

4.1.3 Class diagram

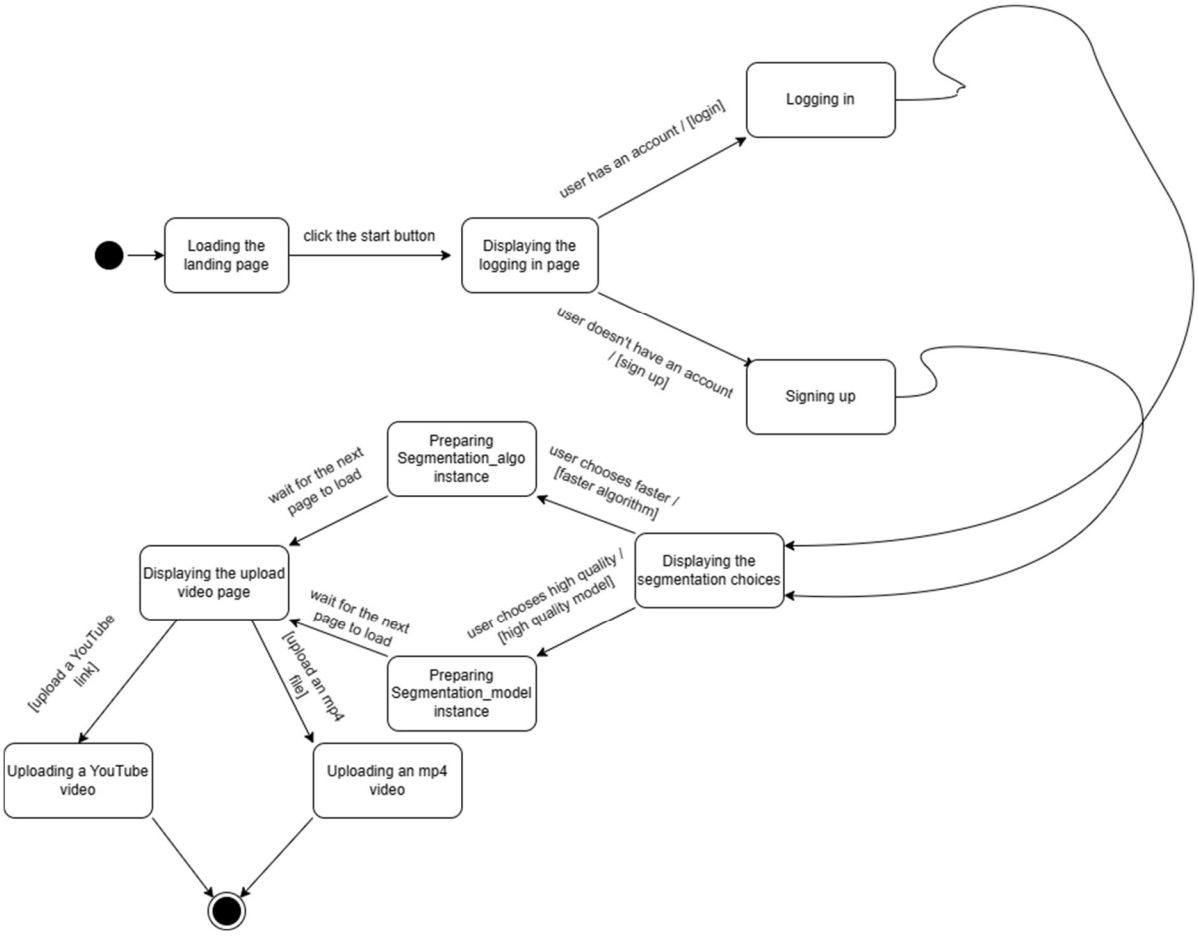
1- **Backend Class Diagram:**



The class diagram above illustrates how the whole background classes works with each other, where most of the AI related classes depend on the ‘‘Server’’ class. The ‘‘Server’’ class is a Flask API where it integrates everything with each other and sends the results of each feature to the Front-end or ‘‘Client server’’ to display the outputs that has been generated by the features.

4.1.4 State transition diagram

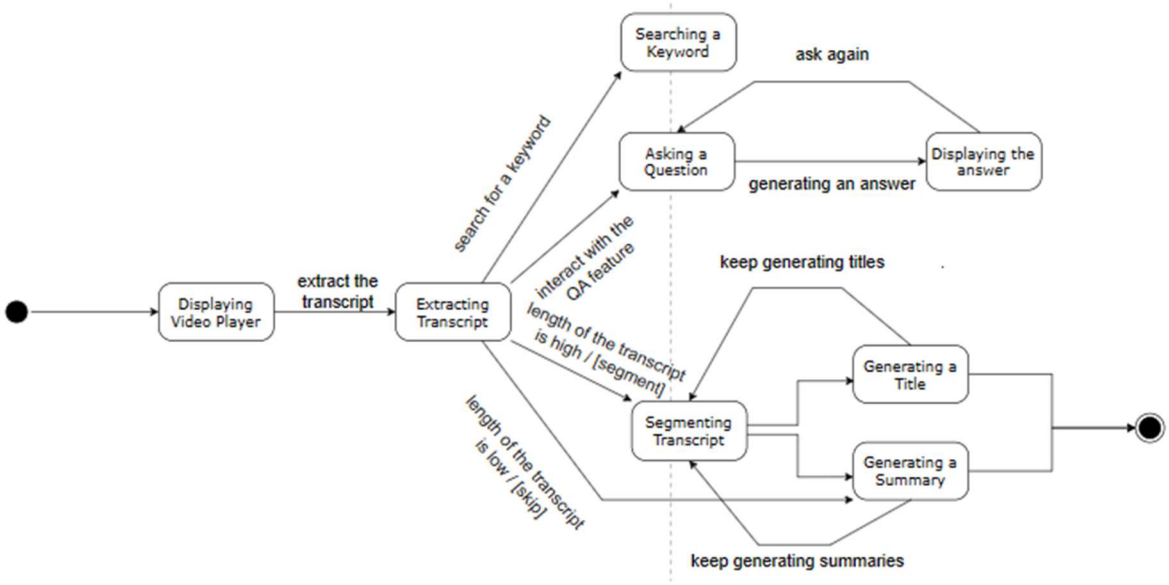
1- **Start state chart diagram:**



This diagram explains the process of what happens before the user reaches the ‘‘Tools’’ page. Firstly, the user enters the website and sees a landing page that tells them to click on the start button. Secondly, the user enters their credentials if they already had an account if not then they need to sign up. Once this process is completed, a web page displays two options for the user to choose whether they want a faster

segmentation algorithm or a high-quality segmentation model. After the user selects their option, another web page will display two options whether the user wants to upload a file from his local computer or paste a YouTube link in order to reach the "Tools" page.

2- **Tool state chart diagram:**

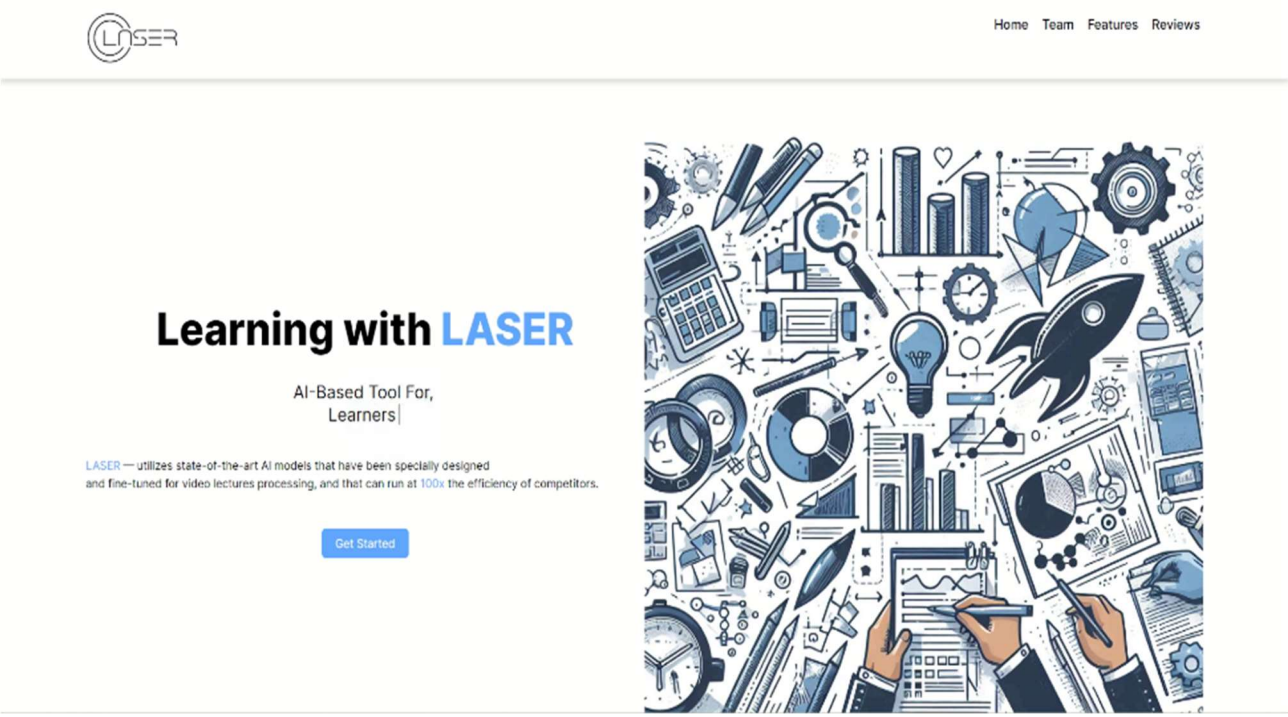


This diagram demonstrates how the process of our main web page which is the "Tool" goes. Once the user pastes their YouTube link or upload one of their mp4 files, the video player will get displayed into its position as assigned. Then, we extract the transcript from the speech and display it on its right position. When the transcript gets extracted, the segmentation part checks weather the length of the transcript is high or low. If it is high, then it generates segments and return an array full of seconds. If not, then it returns and empty list. At the same time the user can search for a keyword in the transcript, and they can start asking questions and wait for an answer from the model that is assigned to do this job based on the transcript. After the transcript got segmented, the summary as well as the titles will get generated for each segment.

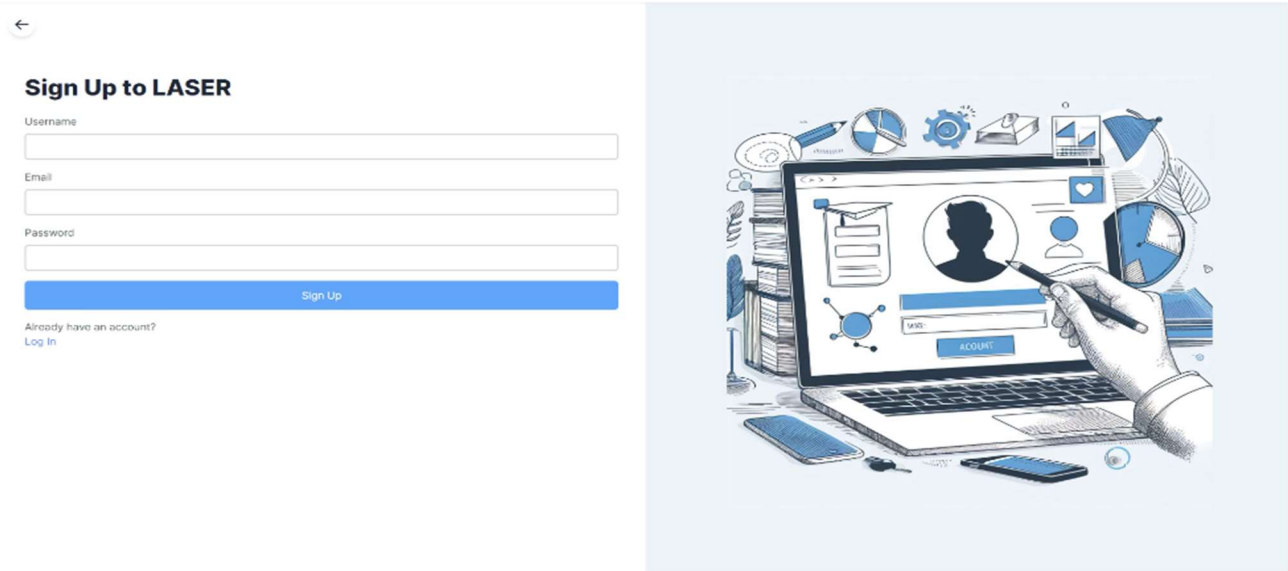
4.2 User interface design (prototype)

In this section, we will be showing some images of our prototype.

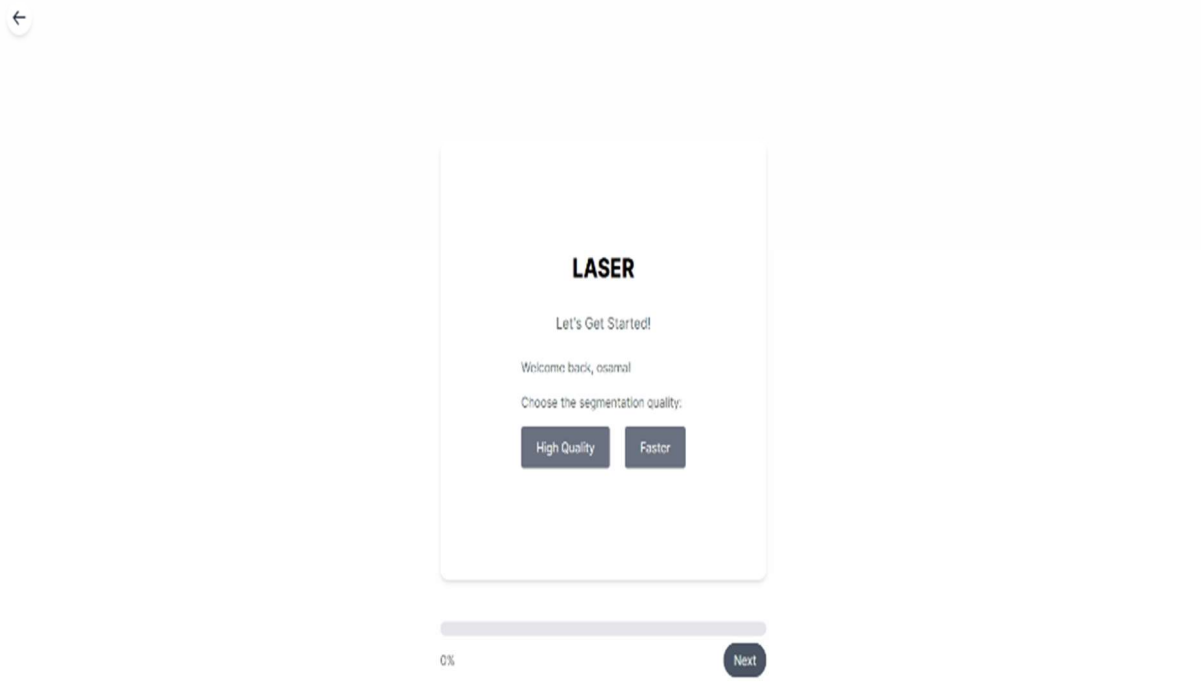
1- Landing page:



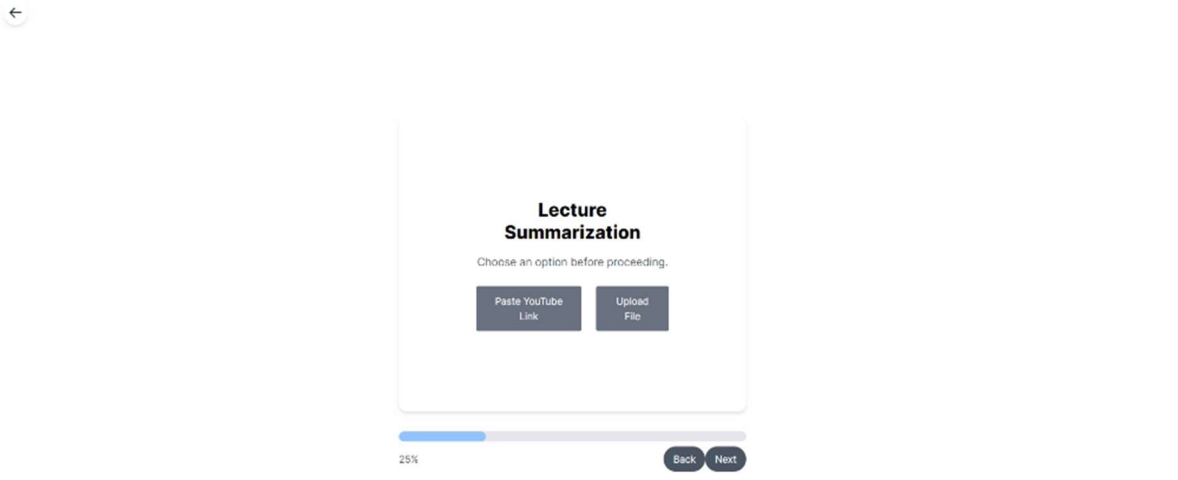
2- Sign-up webpage:



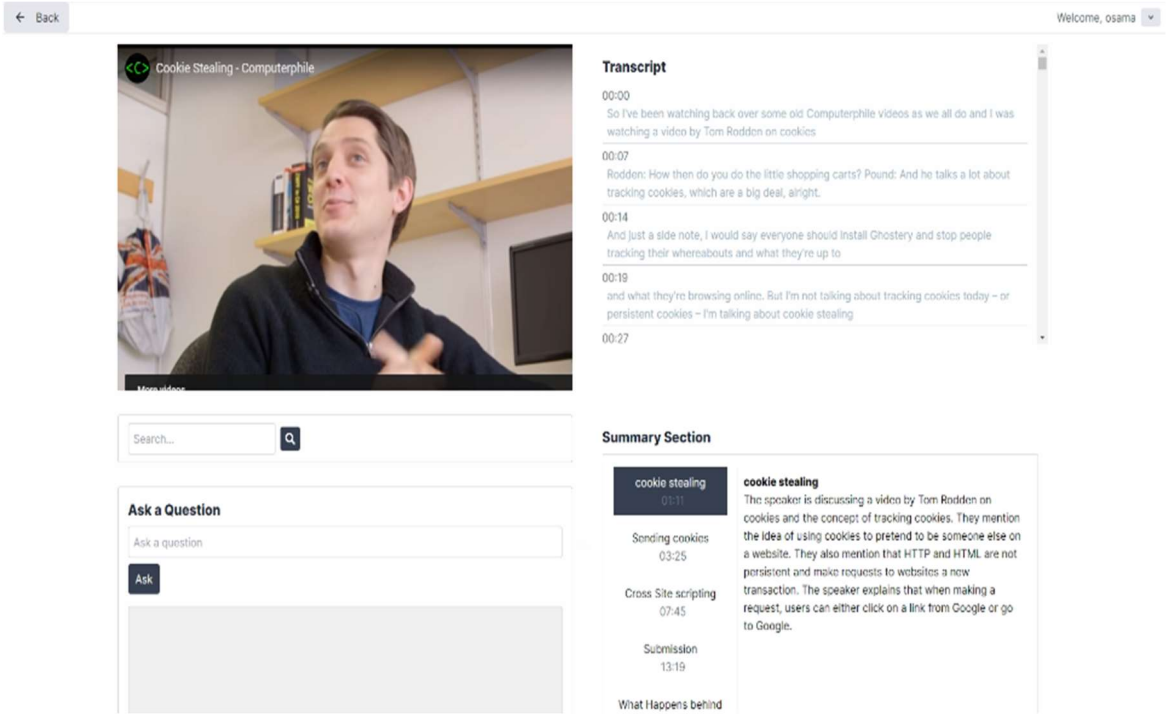
3- <u>Choosing the Segmentation method:</u>



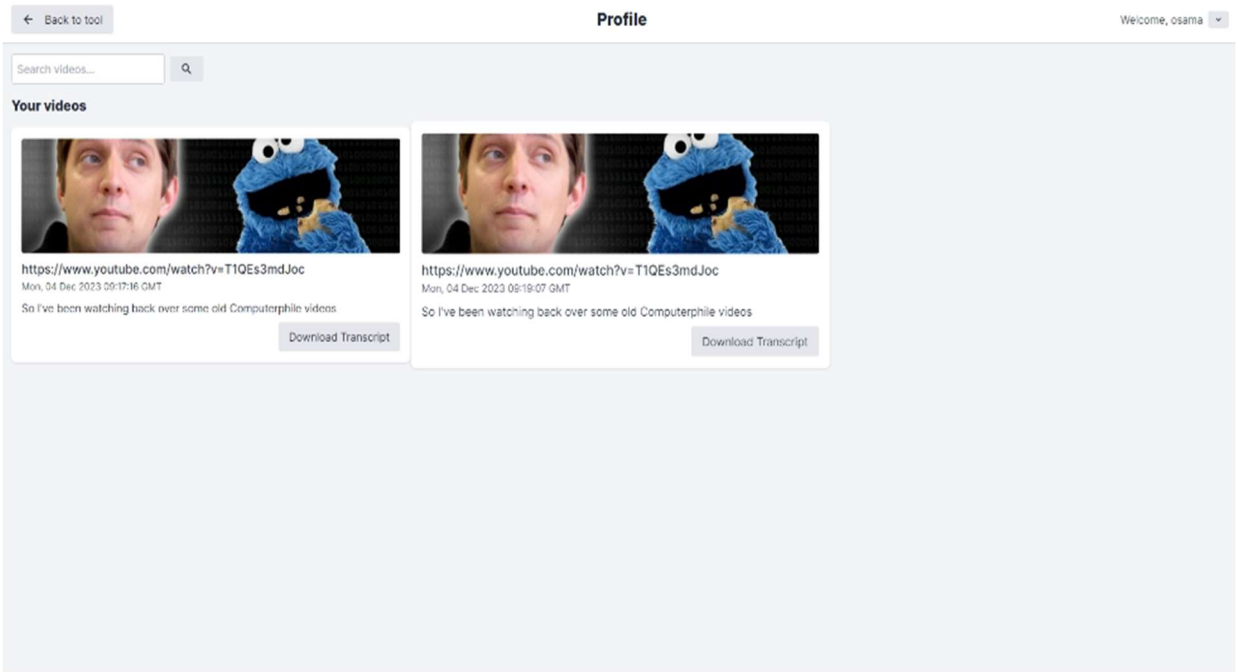
4- <u>Options for pasting a YouTube link or uploading a recording:</u>



5- Tools page:



6- My profile's page:



CHAPTER 5: Implementation Plan

Chapter 5 of the Project Implementation Plan provides an in-depth exploration of the Technical Background within the LASER project. The focal point is the Automatic Speech Recognition (ASR) solution, showcasing the team's comprehensive efforts in deploying and optimizing ASR models. This involves training models on extensive datasets, utilizing deep neural networks, and optimizing for specific hardware. Notably, the integration of the Faster Whisper package enhances processing speed without compromising accuracy. The chapter also details the transition from quantized 8-bit models to full 16-bit models, leveraging the Flash Attention v2 technique for memory efficiency. Challenges in balancing accuracy and processing time for ASR models are discussed, along with the shift to Linux OS for optimized resource utilization. Subsequent sections highlight accomplishments in Lecture Segmentation, Lecture Summarization, and the incorporation of features like embeddings, searching, and Question Answering.

5.1 Technical Background (Tools and Frameworks)

5.1.1 Automatic Speech Recognition

This section of the Project Implementation Plan provides a comprehensive exploration of our ASR Solution, encompassing the complexities of deploying, installing, and seamlessly transitioning the information system into a fully operational state. Throughout this journey, our team encountered various challenges and navigated through a multitude of ASR models to arrive at an optimal solution for the LASER project.

The ASR Solution holds a pivotal role in applications such as transcription services, virtual assistants, and voice-activated systems, significantly enhancing the efficiency and naturalness of human-computer interaction. The nuanced process of ASR implementation involves training models on extensive datasets, leveraging deep neural networks, and optimizing for specific hardware, including GPUs.

In our pursuit of optimization, we strategically integrated the Faster Whisper package, a reimplementation of OpenAI's Whisper model utilizing CTranslate2. CTranslate2 is a C++ and Python library designed for efficient inference with Transformer models. This implementation boasts a remarkable fourfold increase in speed compared to

OpenAI's Whisper, achieving the same level of accuracy while consuming less memory.

The enhanced efficiency of Faster Whisper can be further augmented by employing 8-bit quantization on both CPU and GPU. In our specific case, we leverage GPU utilization to optimize performance. Through rigorous testing, we observed that this model requires less than a minute for transcription on the GPU, utilizing a maximum of 3091MB of GPU memory. It's noteworthy that these metrics are applicable to the Large-v2 model. Comparatively, using the OpenAI/Whisper model for the same task would necessitate over four minutes for transcription and entail a maximum GPU memory usage of 11325MB. The Faster Whisper package has proven instrumental in significantly improving processing speed without compromising the accuracy of the transcription process.

You can find a more detailed comparison between the Faster Whisper and OpenAI's Whisper [31] in the table below:

Table 3. Large-v2 Model on GPU

Implementation	Precision	Beam Size	Time	Max.GPU memory	Max. CPU memory
Openai/whisper	Fp16	5	4m30s	11325MB	9439MB
Faster-whisper	Fp16	5	54s	4755MB	3244MB
Faster-whisper	Int8	5	59s	3091MB	3117MB

Table 4. Small Model on CPU

Implementation	Precision	Beam size	Time	Max. memory
Openai/whisper	Fp32	5	10m31s	3101MB
Whisper.cpp	Fp32	5	17m42s	1581MB
Whisper.cpp	Fp16	5	12m39s	873MB
Faster-whisper	Fp32	5	2m44s	1675MB
Faster-whisper	Int8	5	2m04s	995MB

5.1.2 Whisper with Batching and Flash Attention v2

Later in the project we discovered a better way to utilize the Whisper model. Instead of using quantized 8bits model, we used full 16bits model. However, we loaded the model through the transformers pipelines package [32]. Furthermore, we enabled the Flash Attention v2 technique [33]. This technique enables memory efficient attention, allowing us to greatly reduce the memory footprint of the whisper model.

More importantly, in our case, we were able to enable batching, which means that we batch multiple samples from the audio file and send them to Whisper to be processed at once This model surpassed expectations in terms of accuracy, and greatly increased the transcription speed.

Our journey also involved overcoming challenges with other ASR models. Some models, while providing accurate results, proved excessively slow, significantly impeding processing time for longer videos. Conversely, faster models sacrificed accuracy, introducing frequent inaccuracies and speech filler issues.

Furthermore, the integration of Whisper with the Flask server in our web application posed a unique challenge. Initially, the model's runtime and the prolonged loading time for .mp4 files adversely affected the user experience. This prompted further refinement and optimization of the implementation to enhance overall user satisfaction.

In further consideration of our operational framework, it becomes evident that the integration of additional models on our website, including Summarization models and Question Answering models alongside the Whisper model, has imposed a considerable load on the GPU. This collective demand for computational resources necessitated a strategic shift towards a more efficient operating system.

After careful evaluation, we opted for Linux OS to capitalize on its efficient and modular kernel architecture. This transition was motivated by Linux's reputation for optimizing resource utilization, providing a more robust environment for concurrently handling diverse computational operations.

Moreover, it's essential to note that the Flash attention v2, a critical component in our system, is exclusive to Linux and does not function on Windows. This technical constraint further solidified our decision to adopt Linux as the preferred operating system for our operations. This strategic move ensures the seamless integration and optimal

performance of our various models, enhancing the overall efficiency and stability of our website.

In conclusion, the iterative and adaptive approach undertaken in our ASR implementation, along with the exploration of multiple models and the resolution of encountered challenges, reflects our commitment to delivering a robust and refined technology solution for the LASER project. The integration of transformers and the meticulous adjustment of parameters stand as a testament to our dedication to surpassing the unique demands outlined in the LASER project's site-specific implementation requirements, and transitioning into the Linux OS was essential in order to increase the performance of the website in general.

5.1.3 Lecture Segmentation

In this section, we provide a detailed explanation of what we have accomplished in the segmentation task.

In this task we started first be exploring if there were anyone did this kind of work before. Any research paper or any published code on the internet. During this phase, we have found couple of research papers discussing different parts of the segmentation such as OCR where we extract the main parts of a video and consider them as segments, transcription where we segment based on the transcription of the video, etc.

One of the papers we have found which is called Temporal Lecture Video Fragmentation using Word Embeddings [Temporal Lecture Video Fragmentation Using Word Embeddings | SpringerLink], discusses of how it uses a segmentation algorithm “not AI based” to segment a video into segments based on a transcript. Unfortunately, the code of this algorithm hasn’t been provided with the paper. So, we decided to write the code by ourselves relying on what the paper had discussed. We will be discussing this algorithm in detail in the following subsection.

After the completion of this code, we first ran into a problem where we found the algorithm isn’t that good to generate segments based on evaluation metrics the paper talked about. We found that the problem was with the dataset we used to evaluate the efficiency of the algorithm, we have used a wrong dataset. After we found the issue, the algorithm scores have increased dramatically. With a bit of tuning with some parameters we were able to increase more.

At the same time, we decided to follow a supervised approach to solve the problem. Given that we have the MIT Chapters dataset, which consists of transcripts of lecture video chapters and their corresponding time segments, we could train a classifier on this data. The classifier takes each sentence and decides if it indicates an end of a segment or not. Therefore, this is a sequence labelling task. Our research into this type of problem has shown us that the BiLSTM model has been successfully used before for text segmentation specifically. However, it was not utilized for lecture video transcripts segmentation yet. We decided to use this model and train it on our MIT dataset, and as far as we are aware, we are the first to utilize it for this task.

In conclusion, the BiLSTM model had a slightly higher quality than the segmentation algorithm. But the segmentation algorithm is slightly faster. Therefore, we decided to use both algorithms for the users to make their own choices whether they want a segmentation model that generates more accurate segments or a segmentation algorithm that generates segments faster.

1- Segmentation Algorithm:

The paper that we have mentioned above gives a simple approach to segment the video based on a transcript. The following figure shows the approach of the proposed method, this diagram found in the paper:

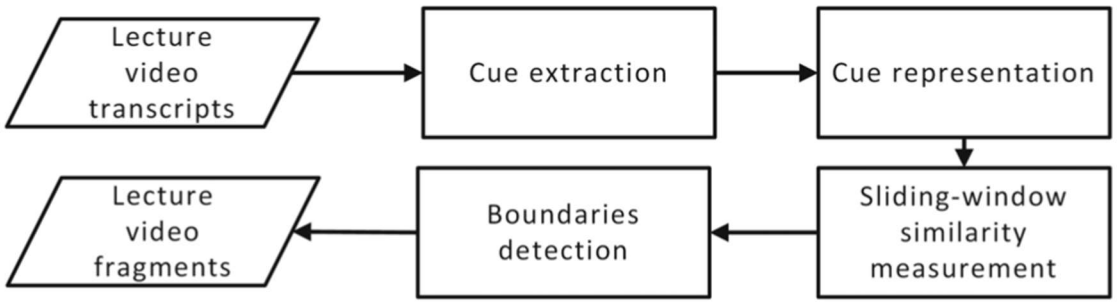


Figure 6. Transcript-Based Video Segmentation Approach

Following points are explanation of each part found in the above diagram:

1. **Lecture video transcripts:** in this part we need to get a transcript so that we continue the method.
2. **Cue extraction:** the paper discusses two approaches. The first approach is to

extract the Noun Phrases from the available text. The 'noun phrase' is basically a noun, plus all the words that surrounds this noun, such as adjectives, relative clauses, etc. The second approach is inspired from the query analysis and decomposition method of an ad-hoc video search (AVS) system.

3. **Cue representation:** it also discusses two different representations. A Bag-of-words approach with an N-gram language model. The second approach is that they utilized Word2Vec, it's a neural-network-based word embedding method that transforms words into a vector.
4. **Sliding-window similarity measurement:** the previous two steps are performed after a sliding window moves across the entire text of transcript with a certain step. After the text has been divided into parts based in the number of certain steps, in each sliding window we follow the process of cue extraction and cue representation to get a vector representation for each sliding window. Finally, the cosine similarity function is utilized to calculate the similarity between two windows.
5. **Boundaries detection:** Following the calculation of the cosine similarity, a 1D signal y = f(x), where x represents the time and the y represents two neighbouring windows similarity score, is produced. Then, the valleys and peaks, which are important variables for the following equation, of the signal are detected. The depth of a valley is calculated based on the distances from the peaks on both sides of the valley. The depth of a valley indicates how big the change in this particular time interval is. If the depth of a valley is high, then we assume that the context of the two windows is not similar therefore this time point is assigned as a boundary.
6. **Lecture video fragments:** in this last part, we obtain the final results.

After we have thoroughly read the paper, we started by implementing the code to solve this kind of problem. Evaluating the algorithm part will be discussed in Chapter 6 of this paper.

5.1.4 Lecture Summarization

In our project we are utilizing the lecture transcript to automatically generate a textual summary. This is considered an automatic text summarization task. Because we want the output summary to have coherent and meaningful sentences, we will be generating an abstractive summary. Abstractive summaries are human-like in the way they paraphrase the summary, in contrast to extractive summaries which are just key sentences that have been selected (Cajueiro et al., 2023). In our work we utilize both techniques, as detailed in previous reports. In this section, we will go over some background on the chosen models, the challenges we faced and our solutions to them, and a detailed explanation of the process of training and using the summarization models in our application.

1- **Abstractive Summarization Pre-trained models**

Our research in the junior phase of the project has concluded that to get the best abstractive summary possible, while retaining maximum efficiency, we must use pre-trained models (Liu & Lapata, 2019). These models have been pre-trained for the summarization task in both unsupervised and supervised manners. In our case, after careful study of the available architectures, including the T5 model (Raffel et al., 2020), PEGASUS (Zhang et al., 2020), and BART (Lewis et al., 2019). We have performed extensive experiments in the junior phase and decided that BART family models offer the best ratio of performance vs. accuracy. Therefore, we have decided to fine-tune BART models and incorporate them into our application.

To ensure maximum performance, we had to choose a good pre-trained checkpoint. On the Hugging Face Models Hub, we found a checkpoint provided by meta that has been fine-tuned on the CNN/DailyMail dataset (Ainize/Bart-Base-Cnn · Hugging Face, 2022), which contains news articles and corresponding summaries. This aligns well with our goal because news articles often contain written human speech. We then fine-tune this checkpoint on our own datasets.

2- Summarizing full lectures vs. lectures chapters

In our project, we aim to summarize both full lecture videos and segmented (chaptered) lecture videos. In the former case, we take the full transcript and generate a summary. This is only feasible with short enough videos due to the maximum length limitations of the BART model. In the latter case, each segment transcript part is summarized individually, and the output is a list of summaries.

With this approach, we overcome the problem of the maximum input (context) length, which is 1024 tokens for base BART models. By simply checking if the video length exceeds this threshold and performing segmentation in this case.

3- Fine-Tuning Datasets

We have fine-tuned our models on two datasets: AK Lectures and MIT Chapters. We used the AK Lectures dataset to obtain a model geared towards summarizing full short-to-medium lecture videos, from start to finish. The MIT Chapters dataset contains lectures segments and their corresponding summaries, generated by GPT-3.5. Therefore, we used it to fine-tune and obtain a model specialized for summarizing lecture transcripts.

4- Long Context Length

Furthermore, we also utilized a different approach to mitigate this problem. For lecture transcripts, window 1024 is too short. Most lecture videos more than 5 minutes in length will contain more tokens in their transcript. We have looked for solutions other than segmenting the transcript and found two options. We have tested these two options and outlined our results in detail in the progress report. We summarize our findings here:

1. Compressing the transcript with extractive summaries.
    a. This option utilizes TextRank (Mihalcea & Tarau, 2004), LSA (Kireyev, n.d.), or BERT (Liu, 2019) to compress the transcript, then it passes the extractive summary into the abstractive BART model.
    b. Fast, but less quality.
    c. Experiments concluded the resulting abstractive summaries have less ROUGE scores.
2. Using the Local, Sparse and Global attention (LSG)

a. This method improves the efficiency of the Attention mechanism in the BART model by making it linear.
b. Less memory usage, faster inference.
c. Almost the same or better quality.
d. Experiments concluded the resulting abstractive summaries have better ROUGE scores than the extractive summaries.

Considering the above, we have decided to additional train a LSG version of BART to summarize short-medium length videos. We could have gone up to 16384 tokens in length, but due to GPU memory constraints, we went for 2048 tokens.

With this setup, if a video is longer than 10 minutes (Which corresponds to about 2000 tokens), we perform segmentation. Otherwise, if it contains less than 1024 tokens, we use the base BART model. If it falls between 1024 and 2048 tokens, we use the LSG BART model.

For longer videos that are segmented, we summarize each individual transcript with the base BART model. If one segment is longer than 1024 tokens, we compress it using TextRank algorithm before passing it into the BART model. We do this to reduce the computational power and GPU memory requirements during inference.

5- **Segment Title Generation**

When generating a summary for a section, we also wanted to obtain a short title. This is later used in the application for quick navigation. Our MIT Chapters dataset contains human written titles for each chapter; therefore, we utilized it for training and evaluating title generation schemes. We have experimented with the following methods for title generation:

1. Fine-tuning a Seq2Seq transformer from scratch on the MIT Chapters.
    a. Evaluation showed repetitive titles and non-meaningful titles sometimes.
2. Extracting keywords with pre-trained embeddings and using TF-IDF heuristics.
    a. Multiple keywords but not meaningful.
3. Extracting key topics with BERTopic and using TF-IDF hearistics.

    a. Not coherent topics and not varied enough.

4. Fine-tuning the BART-base model for both title and summary generation at the same time.
    a. Generates coherent titles that closely match our dataset.
    b. Achieves a 62% ROUGE score.

Our last method involves generating a title and a summary in one forward iteration. The model is trained on summaries that include the title in the first line. Therefore, it learns to always output the title in the first line, and the summary in the subsequent lines. This has not only solved the title generation challenge, but also greatly improved our summarization model ROUGE scores from 58% to 62%.

5.1.5 Embeddings

For the AI models or the computer to understand text, the text should be conveyed to various numbers. This is where Embeddings come from. We have used embeddings in various parts in order to accomplish our project. Where we have used it to embed the sentences that extracted from a transcript for further use such as segmenting the recordings, answering based on the users' questions, etc. The two embeddings that we were able to use are flag Embedding [34] and distillbert-based [35].

The following points discussed where we have used embeddings in our application:

- For the segmentation feature, we have used a fast-Embedding model to embed the sentences that got extracted from the transcript.
- For the Question and Answering feature, we have used a SentenceTransformer model called distilbert-based to get the top k sentences that matches the questions to pass those k sentences with the question for the model to generate answer accordingly.
- Searching my lectures: When a user enters a prompt (keywords, phrase, or a question) in the search box of the "my lectures page", we use the embeddings models to perform semantic search and retrieve most relevant lectures.

To integrate embeddings with our application and for the main features of the application to use it, we had to build a class called ‘‘Embeddor’’ and integrate it with the ‘‘Flask server’’ so the application can use it smoothly and easily.

5.1.6 Searching

We have added an additional incredible feature in our LASER. Which will help the user a lot in navigating back to their old recordings. We called it Searching.

After the users create an account and use it for their educational purposes, for each recording they upload or paste a YouTube link LASER will automatically save their recordings in their profiles. So, the user can easily return to their old recordings if they want to fetch something up. There is a profile page for each user who created an account. There they will see a list of recordings that they have uploaded before. They can search there for whatever recording they uploaded before. Also, there is an option to download a pdf file that has the summaries that have been generated from our website as well as the titles if the recording got segmented.

We believe that by adding this feature, the users will easily return to their old recordings for further work, and this will help them a lot instead of uploading the same recording multiple times.

5.1.7 Question Answering

One of the most amazing features in our application is the Question and Answering feature. We have found that by adding this feature to our application will help our users to understand their recordings more easily and improve their experience with the application. We have tried many different models that generate a great answer and a faster response to the user. We found that llama2 [36] is the best model to accomplish our desired task.

The implementation details are simple. The server waits for a request from the user or a ‘‘question’’ to provide. Once the user submits their question, the server passes this question along with the transcript to generate an answer for the user. The Question and Answering class first get the top k sentences from the transcript matching the question and then passes those two parameters to the model to generate an answer accordingly.

We have built a Question and Answering class called ‘‘QuestionAnswering’’ to integrate it with our application. So, the user can be able to use this feature to help them in understanding more their recordings.

5.2 LASER web application

The LASER (Learning from Automatically Summarized and Segmented Educational Recordings) web application is a comprehensive tool designed to enhance educational video content by providing functionalities such as segmentation, transcription, summarization, and question-answering for students, educators, and everyone. The successful implementation of this project required the integration of various programming languages and technologies. This section outlines the key components involved in the LASER web application and details their specifications. As depicted in Figure 7, further elaboration on the three-tier application system architecture is provided in the subsequent subsections.

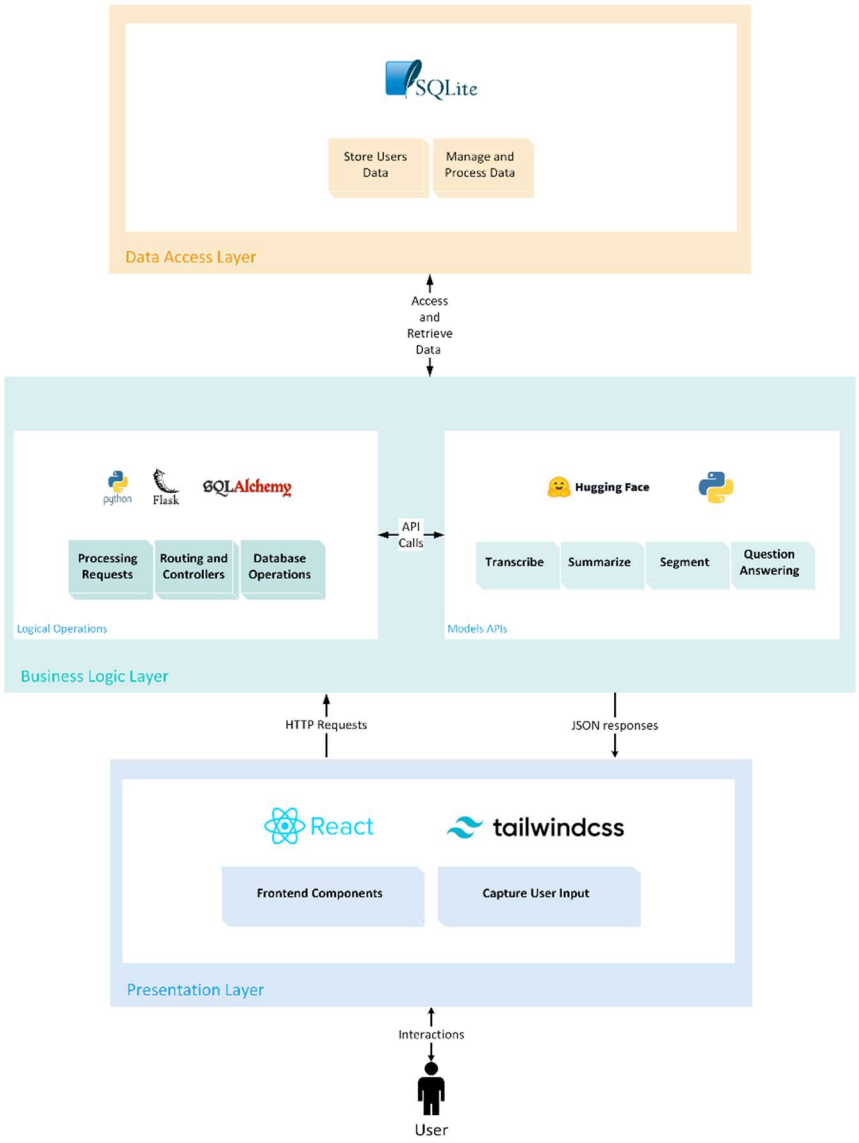


Figure 7. Illustration of the Three-Tier Application System Architecture

5.2.1 Web Server (Flask)

The web server component of the laser application is built on flask, a micro web framework for python renowned for its simplicity, flexibility, and functionality. in the initial stages of development, our team extensively evaluated various frameworks and tools, considering Django during the junior phase. However, after meticulous testing and consideration, flask emerged as the optimal choice due to its lightweight nature and user-friendly design. flask's modular architecture facilitates the seamless integration of various components of the application, allowing for efficient handling of HTTP requests and responses. Its simplicity doesn't compromise functionality, and it provides a clean and organized structure for the laser web server. A comparison between the two different components is stated in Table 5.

Table 5. Comparison Between Django and Flask Frameworks

Aspect	Django	Flask
Type	Full-featured framework	Micro-framework
Complexity	More complex	Simpler
Functionality	Greater functionality	Basic functionality
Performance	Can be sluggish	Lighter and faster
Control	Takes care of a lot of heavy lifting	Gives complete control on a smaller scale

The selection of Flask for the laser application stemmed from its lightweight nature, user-friendly design, and modular architecture, proving ideal for the project's needs. Leveraging Flask's extensive ecosystem and vibrant community support played a crucial role in constructing the laser web server, ensuring scalability and maintainability through a rich array of available extensions and plugins. Moreover, Flask's support for URL routing and template rendering significantly streamlined development, enabling a focused implementation of core features such as segmentation, transcription, summarization, and question-answering within the laser application. This underscores the Flask-based web server's status as the backbone of the project, showcasing the framework's remarkable flexibility and functionality despite its simplicity.

5.2.2 Client Server (React)

The client-side of the laser web application is constructed using React, a powerful JavaScript library for building user interfaces. React's component-based architecture is a foundation stone of the LASER user interface, allowing the creation of modular and reusable interface elements. This approach not only enhances the maintainability of the codebase but also fosters a responsive and interactive user experience. Leveraging React's virtual DOM (document object model) ensures efficient rendering of UI components, contributing to the overall performance optimization of the laser application. As depicted in Figure 8, the popularity of React has shown steady expansion. The associated Google Trends chart highlights a swift rise in popularity, aligning with the continuous integration of new features over an extended period [37].

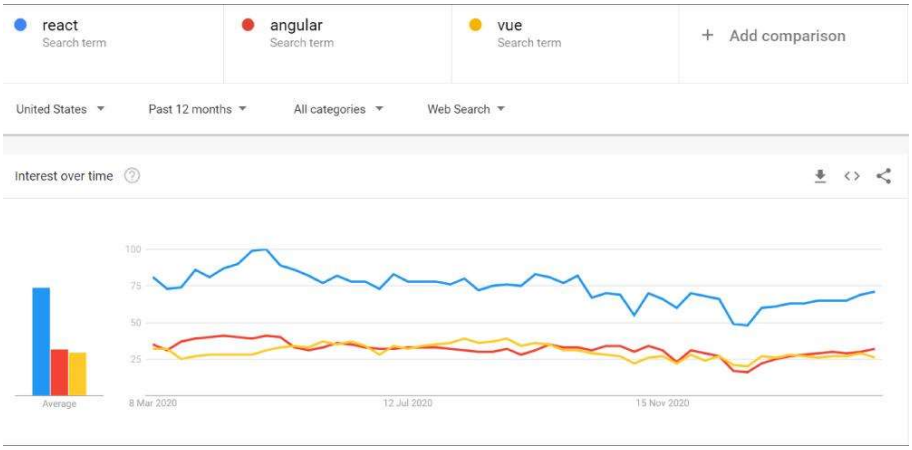


Figure 8. Evolution of React: Popularity and Feature Expansion Over Time

The client-server interaction is organized seamlessly through restful API endpoints establishing a dynamic and intuitive interface for users to interact with the laser educational tools. React's ability to manage the state of components efficiently ensures a fluid and real-time experience for users engaging with educational content. Figure 9 shows the modular nature of react components as it aligns with the diverse functionalities of LASER, enabling a clear separation of concerns and facilitating the integration of new features in the future [38].

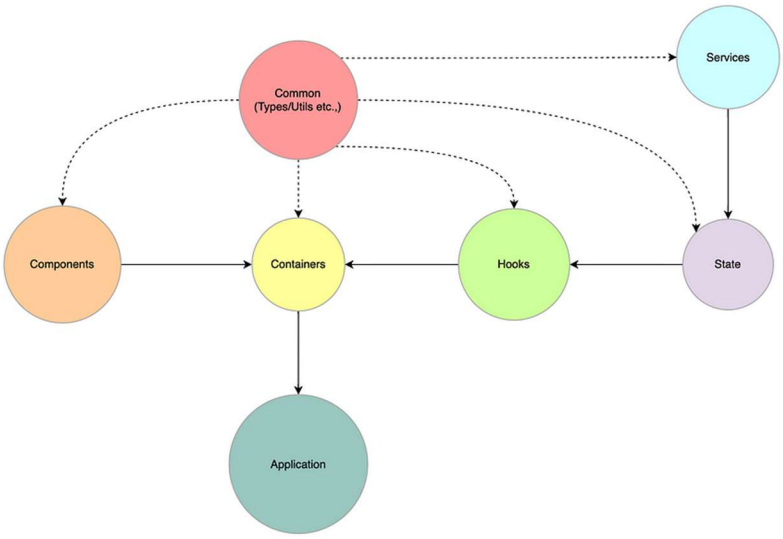


Figure 9. Modular Architecture for React Applications

1- Frontend-Backend Connection (Axios, CORS, Flask Endpoints)

The connection between the frontend and backend of the LASER web application is facilitated by Axios, a popular, promise-based HTTP client for the browser and Node.js. Axios provides a simple and clean API for sending HTTP requests from the React-based frontend to the Flask-based backend. It supports a wide range of HTTP request methods, including GET, POST, DELETE, and PUT, which are essential for the RESTful API endpoints of our application.

The use of Axios in our project was not without challenges. We encountered several issues related to Axios, which are common among many users of React and Flask. These issues were primarily related to Cross-Origin Resource Sharing (CORS), a mechanism that allows many resources (e.g., fonts, JavaScript, etc.) on a web page to be requested from another domain outside the domain from which the resource originated.

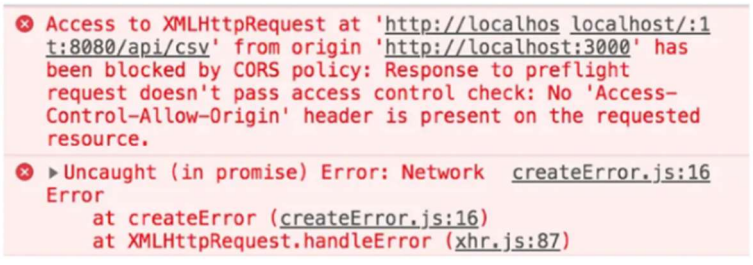


Figure 10. Example of CORS Errors in React [39]

To overcome these challenges, we implemented CORS on our Flask server, allowing it to accept requests from the React client running on a different origin. This was achieved by using the Flask-CORS extension, which is a simple, open-source Flask extension for handling CORS. It supports simple and complex requests, including those with credentials.

Debugging the issues related to Axios was a significant part of our development process. We used various tools and techniques for debugging, such as logging, breakpoints, and network inspection tools. Through this process, we gained a deeper understanding of how Axios works and how to effectively use it in our application.

2- <u>Debugging and Error Handling</u>

Debugging and error handling were crucial aspects of our development process. We encountered various errors during the development of the LASER web application, particularly related to Axios and Flask endpoints. These errors were often due to misconfigurations, incorrect data types, or network issues.

To handle these errors, we implemented robust error handling mechanisms in our code. We used try-catch blocks to catch errors and exceptions, and we logged these errors for further analysis. We also used debugging tools provided by our code editor and browser to inspect the state of our application and track down the source of errors. Through this debugging process, we were able to identify and fix issues in our code, leading to a more stable and reliable LASER web application. This experience also taught us valuable lessons about the importance of thorough testing and error handling in software development.

5.2.3 AI Integration

The AI integration component of the LASER web application is a critical aspect that empowers the platform with advanced capabilities such as automatic transcription, segmentation, summarization, and question-answering. Our goal was to seamlessly integrate state-of-the-art AI models into the LASER framework, enhancing its educational functionalities. This section provides insights into the integration process of the AI components.

1- **Automatic Speech Recognition (ASR) Model**

To enable the automatic transcription feature, we integrated a robust Automatic Speech Recognition (ASR) model into the LASER web application. The ASR model is responsible for converting spoken language into written text, thereby facilitating the creation of accurate transcripts for educational videos.

During the initial phases, we faced a challenge with the compatibility of our ASR model with various audio file formats. Specifically, the model initially only supported WAV files, which posed limitations on the types of files users could upload. Our ongoing efforts involved adapting the ASR model to handle WAV files after converting them from the MP4 file format. This approach aims to enhance user convenience.

2- **Summarization and Segmentation Models**

The LASER web application incorporates advanced machine learning models for summarization and segmentation. These models play a pivotal role in automatically extracting key information from educational videos, offering users concise summaries and segmented content.

Our team extensively researched and implemented cutting-edge algorithms for text summarization and video segmentation. These models leverage natural language processing (NLP) to analyze and distill relevant information from educational content. The integration of these AI models enhances the educational experience by providing users with efficiently summarized content and well-organized segmented videos.

3- **Question-Answering Model**

The LASER web application's question-answering functionality relies on a sophisticated Question-Answering (QA) model. This AI component enables users to interactively engage with educational content by posing questions related to the material.

Our approach involved integrating a QA model capable of understanding and extracting relevant information from the transcripts generated by the LASER platform. This ensures that users receive contextually relevant answers to their queries, fostering a dynamic and interactive learning experience. Figure 11 shows a simplified overview of the system with the integrated components.

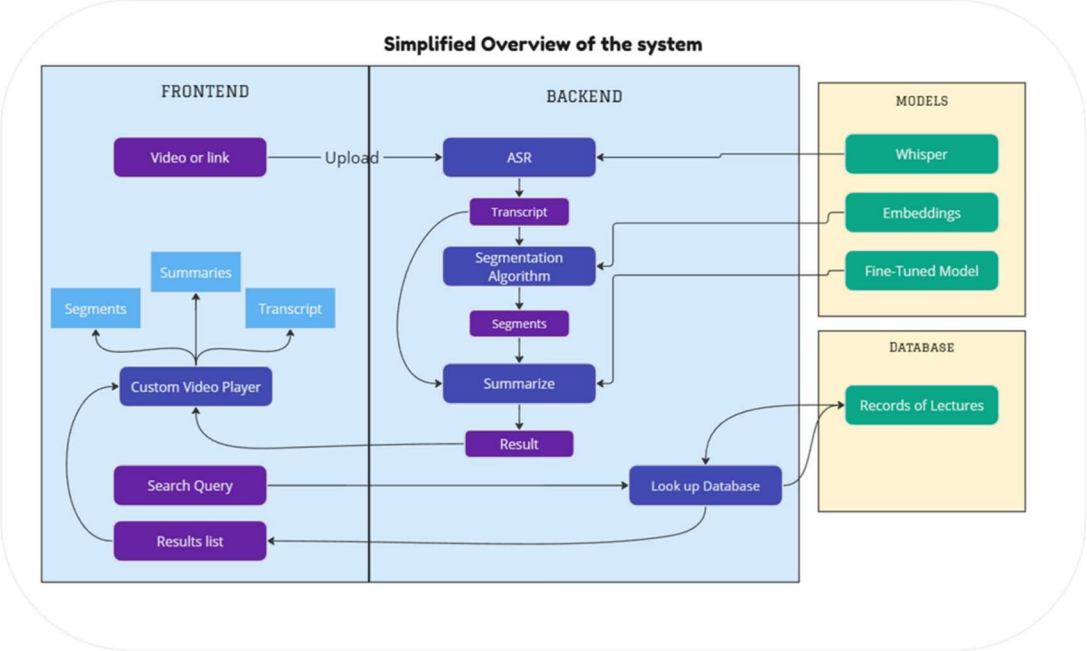


Figure 11. Overview of Our LASER System Components

5.2.4 Database

The LASER web application relies on a robust database system to store and manage data efficiently. For our project, we chose SQLite as the relational database management system, coupled with the SQLAlchemy library for seamless integration with our Flask-based web server. SQLite's lightweight and serverless nature made it an ideal choice for our educational platform. The database design accommodates the storage of user information, video metadata, transcriptions, and AI-generated summaries. The utilization of SQLAlchemy further simplified database operations, providing an object-relational mapping (ORM) system that abstracts database interactions into Pythonic code.

5.2.5 Testing and Quality Assurance

During the development lifecycle of the LASER web application, rigorous testing procedures were employed to ensure the reliability and functionality of the platform. Several software tools were instrumental in our testing and quality assurance processes.

1- **Postman for API Testing**

Postman served as a pivotal tool for testing the functionality and reliability of our RESTful API endpoints. Through Postman, we systematically crafted and executed API requests, validating the correctness of data retrieval, data creation, and overall, API behavior. This facilitated comprehensive testing of the communication between the front-end and backend components, ensuring a robust and error-free interaction.

2- **DB Browser for Database Inspection**

DB Browser emerged as a valuable asset for inspecting and managing our SQLite database. This tool allowed our team to visually explore the database structure, inspect tables, and verify data integrity. DB Browser facilitated efficient debugging of database-related issues and provided insights into the intricacies of our data storage, contributing to the overall reliability of the LASER web application.

5.3 Implementation Details

5.3.1 Whisper ASR Service

As detailed in the technical background section, we have decided to move the ASR processing into another device that runs on Linux, allowing us to utilize the efficiency of the Flash Attention V2 package while reducing the load on our main GPU. In order to do this, we had to create a service that runs continuously on the other machine, and that provides a port for access from the main machine. Furthermore, it had to listen to requests for transcript, perform the transcription with the Whisper model, and return the result.

1- **BentoML**

Our research for a solution to this problem led us to BentoML [40]. This is a platform to deploy AI solutions at scale. It allows us to package our required models into ‘bentos’, which store all the models weights and meta data. Furthermore, it provides a way to easily create multi-threaded code to run more than one instance of a model.

In our case, we deployed two runnable instances running Whisper ‘base’ model. This way we can support up to 2 transcriptions simultaneously. The following code snippet is the full code required to run the service.

```python
import bentoml
from bentoml.io import File, Text, JSON
import io
import torch
import uuid
import os
from transformers import pipeline

class WhisperRunnable(bentoml.Runnable):
    SUPPORTED_RESOURCES = ("nvidia.com/gpu", "cpu")
    SUPPORTS_CPU_MULTI_THREADING = True

    def __init__(self,bentoml_pipe,batch_size=16) -> None:
        self.pipe= pipe = pipeline("automatic-speech-recognition",
                bentoml_pipe,
                torch_dtype=torch.float16,
                model_kwargs={"use_flash_attention_2": True},
                device="cuda")
        self.batch_size=batch_size

    @bentoml.Runnable.method(batchable=True, batch_dim=0)
    def transcribe(self, audio):
        outputs = self.pipe(audio,batch_size=self.batch_size,chunk_length_s=30,return_timestamps=True)
        return outputs


base_pipe = bentoml.transformers.get("whisper-base.en:latest")
small_pipe = bentoml.transformers.get("whisper-small.en:latest")
whisper_base_runner = bentoml.Runner(WhisperRunnable,name="whisper_base_runnable",
runnable_init_params={"bentoml_pipe":"openai/whisper-base.en","batch_size":24},
models=[base_pipe])
svc = bentoml.Service("whisper_asr", runners=[whisper_base_runner])


def generate_unique_filename(extension=''):
    unique_name = str(uuid.uuid4())
    if extension:
        return f"{unique_name}.{extension}"
    return unique_name

@svc.api(input=File(), output=JSON())
async def transcribe(data):

    #audio = convert_byte_audio(data)

    file=io.BytesIO(data.read())
    file_path = generate_unique_filename('mp4')
    with open(file_path, 'wb') as f:
        f.write(file.getvalue())
    result = await whisper_base_runner.async_run([file_path])
    # Delete the file after processing
    os.remove(file_path)
    return {'text': result[0]["text"], 'chunks':result[0]["chunks"]}
```

Figure 12. Snippet From the Code Required to Run the Service

5.3.2 Segmentation Algorithm

```
class Segmentation_algo(Segmentation_main):
    def __init__(self) -> None: ···

    def reset(self, video_duration): ···

    def extract_sentences(self, sentences): ···

    def calculate_similarities(self): ···

    def segment_the_transcript(self, transcript_chunks, video_duration): ···

    def calculate_the_depth_values(self, cosine_similarities_vals, valleys, valleys_array, peaks):

    def get_the_fragments(self, depth_val, threshold): ···

    def calculate_threshold(self, cosine_similarities_vals, valleys): ···

    def calculate_the_valleys_and_peaks(self): ···
```

Figure 13. Code Snippet of the Class of the Segmentation Algorithm

The above code snippet is the class of the segmentation algorithm, or we can call it Sliding window algorithm. The main part of the code is the calculate_the_depth_values() function. Where after we calculate the similarities between the windows, we use the formula that has been discussed in the paper to calculate the depth of the values. The segment_the_transcript() function combines all the other methods into a single function so it can be used easily when the transcript gets generated, and the user chooses his segmentation option.

5.3.3 Segmentation Model

```
class SegmentationModel(Segmentation_main):
    def __init__(self): ···
    def setup_model(self): ···
    def segment_the_transcript(self,transcript, fragment_frequency = 'normal'): ···
    def extract_window_embeddings(self, sentences): ···
```

Figure 14. Code Snippet of the Segmentation Model

The above code snippet is the class of the segmentation model, where we have used BiLSTM model. The segment_the_transcript() function is the main part of the code where it first embeds each sentence of each window then uses the BiLSTM model to predict the segments.

5.3.4 Summarization

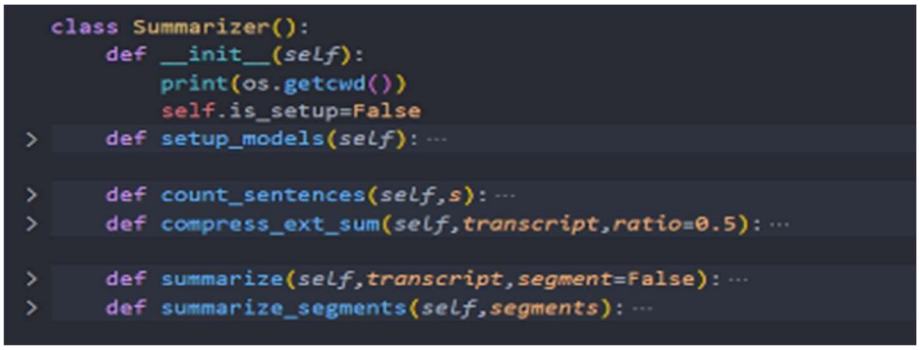


Figure 15. Code Snippet of the Summarization Task

The above code snippet is the class of the Summarization task. There are two functions that summarize the transcript one is called the summarize() and the other called summarize_segments().

5.3.5 Embeddings

As discussed above in the AI technical background section, we have used two types of embeddings in various features. The following figure shows the class of the Embeddor where it uses the distilbert-base. The fast-embedding model has been used for the Segmentation techniques.

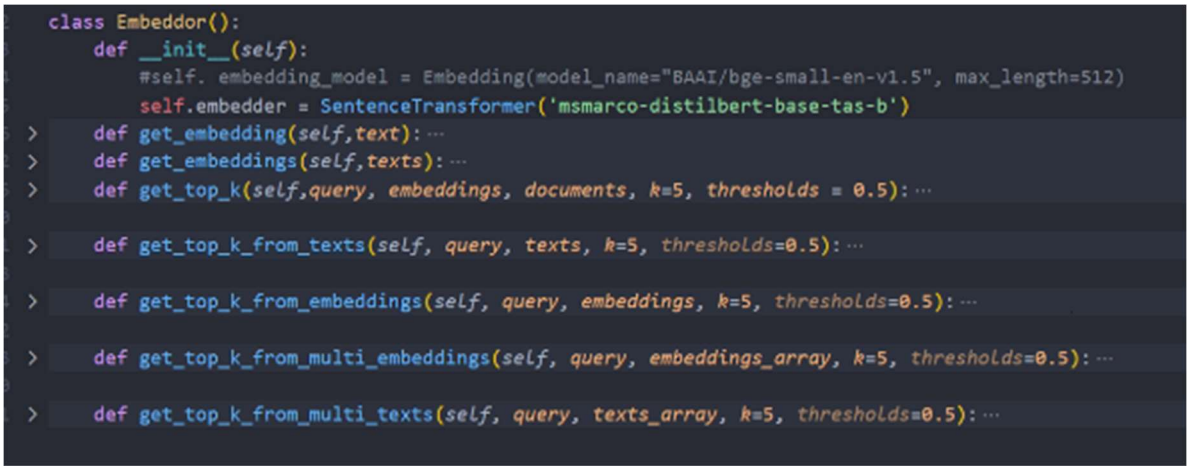


Figure 16. Code snippet of the Embeddings Model

The above code snippet is the class of the embeddings model. The main concept behind creating this class is to use it for our different features such as Question and Answering model. QA model uses the get_top_k_from_texts() function to get the top k sentences from the transcript that matches the query.

5.3.6 Question Answering

```
class QuestionAnswering:
    def __init__(self): ...

    def chunk_the_transcript(self, transcript): ...

    def embed_the_chunks(self, text_chunks, list_of_sentences, question): ...

    def create_prompt(self): ...

    def generate_answer(self, qa_prompt, vector_store, query): ...

    def get_answer(self, transcript, question): ...
```

Figure 17. Code Snippet of the Question and Answering

The above code snippet is the class of the Question and Answering part, where we used llama2 as our model to generate answers for users' questions. The main part in this code is where we create chunks of size 100 each then we embed those chunks by getting the top k sentences that matches the query. Then, the model can generate an answer based on those sentences.

CHAPTER 6: Experiments and Testing

6.1 AI Models Evaluation

In this section, we describe the training and evaluation procedures and results of our AI models for transcription, segmentation, and summarization.

6.1.1 Whisper Transcription

We wanted to assess the accuracy of the generated transcripts of the Whisper 'base' model that we have incorporated in our application. To do so, we needed some reference transcripts and their corresponding audio files. Then, we simply go over each audio file, generate the transcript using Whisper, then compare the generated transcript to the real human-written reference transcript.

From the MIT Chapters we could easily obtain these reference transcripts and audio files, which are from real MIT Lecture videos, and are exactly the type of content we want to support in our application.

To compare between the generated and reference transcripts, we have utilized a vector similarity measure. Usually, the Word-Error-Rate is used instead. However, in our case, the reference transcripts contain many words that are not written. For example, the name of the professor is usually mentioned at the beginning of each sentence, or perhaps the name of students or other speakers. Also, some inaudible words are written as [INAUDIBLE]. Lastly numbers are sometimes written in English and sometimes in decimal, which introduces noise into the evaluation.

On the other hand, the vector similarity metric is based on AI embeddings, and it finds the meaning of each transcript in full and compares it. For completeness, we also show the WER results. Over 7 lecture videos, the average similarity score is 0.99735875. The Word-error-rate we got is 0.363, which is not good enough, but considering the above remarks, we conclude that the WER is low due to discerpancies between the reference transcripts and the speech signals, and not because of the low accuracy of the model.

Table 6 Other metric results for Whisper ASR evaluation

Word Error Rate	**0.363**
Match Error Rate	0.317
Character Error Rate	0.110

```
results = {'actual':[],'whisper':[]}
for transcript, audio in tqdm(zip(transcripts, audio_files)):
    segments, info = model.transcribe(audio, beam_size=5)

    transcripts = []
    for segment in segments:
        #print("[%.2fs -> %.2fs] %s" % (segment.start, segment.end, segment.text))
        #print(segment.text)
        transcripts.append(segment.text)
    whisper_transcript = ' '.join(transcripts)
    cur_wer = wer(transcript, whisper_transcript)
    cur_mer = mer(transcript, whisper_transcript)
    cur_wil = wil(transcript, whisper_transcript)
    cur_wip = wip(transcript, whisper_transcript)
    cur_cer = cer(transcript, whisper_transcript)
    tot_wer += cur_wer
    tot_mer += cur_mer
    tot_wil += cur_wil
    tot_wip += cur_wip
    tot_cer += cur_cer
    results['actual'].append(transcript)
    results['whisper'].append(whisper_transcript)
    #print(transcript)
    #print(whisper_transcript)
    #print(f'WER: {cur_wer} MER: {cur_mer} WIl: {cur_wil} WIP: {cur_wip} CER: {cur_cer}')

print(f'Avg WER: {tot_wer/args.instances}')
print(f'Avg MER: {tot_mer/args.instances}')
print(f'Avg WIL: {tot_wil/args.instances}')
print(f'Avg WIP: {tot_wip/args.instances}')
print(f'Avg CER: {tot_cer/args.instances}')

df = pl.DataFrame(results)
df.write_csv(f'whisper_{args.model_size}_transcripts.csv')
```

6.1.2 Lecture Segmentation

In this section, we present the results of our two segmentation techniques, after evaluating them on the same MIT Chapters dataset. In these experiments, we embed each window of transcript sentences using the fast Embedder. Then, these embeddings are sent for either the algorithm or used as training data instances for the BiLSTM model. To ensure fair evaluation, we show the results when we used a window size of 5, which we found to provide the best results across both methods.

1- Sliding Window Algorithm

Returning to the segmentation algorithm that we have discussed the implementation details earlier; we have used the evaluation metrics that has discussed in the paper to evaluate the algorithm that we have implemented. The evaluation metrics is simple where we need to get the Precision and Recall to calculate the F1-score. Based on the evaluation metrics we have used on MIT lectures dataset; we got our first F1-score result which was around 0.022… This means how the algorithm is bad to predict the segments value. We then saw that the data we have used was not the right dataset for the evaluation metrics since there was many values missing. Also, we tried another solution where instead of extracting the noun phrases in each sentence we extract the sentence as whole and use the fast Embedder model, which has been discussed in Chapter 5 in the Embeddings section. After we changed our code accordingly, we got much better results where the F1-score reached 0.24… Then we kept trying to increase the result by modifying the parameters such as the k variable which is a fixed number of valleys, windows sizes, and m multiplier. The following results are the best results we got after modifying the parameters:

```
Window size = 5
==============================

K value: 3
Precision: 0.6723512336719883
Recall: 0.20386413819874927
F1 Score: 0.19564556447137826
------------------------------------------------

K value: 5
Precision: 0.9273101112723765
Recall: 0.2951110891815878
F1 Score: 0.35083162933041206
------------------------------------------------

K value: 7
Precision: 0.9137155297532673
Recall: 0.35784904223036873
F1 Score: 0.4143515707175556
------------------------------------------------

K value: 10
Precision: 0.8312898841200703
Recall: 0.42415791813840215
F1 Score: 0.45640335733095505
------------------------------------------------
```

```
Window size = 10
==============================

K value: 3
Precision: 0.6208272859216255
Recall: 0.2093712379186296
F1 Score: 0.1852176600863654
------------------------------------------------

K value: 5
Precision: 0.966981132075473
Recall: 0.30887540145393666
F1 Score: 0.38976911376170015
------------------------------------------------

K value: 7
Precision: 0.8545839380745007
Recall: 0.38015611858405807
F1 Score: 0.4601518448371307
------------------------------------------------

K value: 10
Precision: 0.6314102564102545
Recall: 0.45769784628773746
F1 Score: 0.45842596936552804
------------------------------------------------
```

```
Window size = 15
================================

K value: 3
Precision: 0.6973875181422351
Recall: 0.21432432474600063
F1 Score: 0.20062447965775423
------------------------------------------------

K value: 5
Precision: 1.0015723270440284
Recall: 0.31247280556158097
F1 Score: 0.40002293490600793
------------------------------------------------

K value: 7
Precision: 0.8957063376874689
Recall: 0.3852400387485931
F1 Score: 0.46052360046255725
------------------------------------------------

K value: 10
Precision: 0.6959430506600299
Recall: 0.4574340622105895
F1 Score: 0.465037679107678
------------------------------------------------
```

```
Window size = 20
================================

K value: 3
Precision: 0.6560232220609579
Recall: 0.2108828404502578
F1 Score: 0.18969513678631156
------------------------------------------------

K value: 5
Precision: 0.9551282051282054
Recall: 0.3010277140695005
F1 Score: 0.3631038275978987
------------------------------------------------

K value: 7
Precision: 0.8522738268021267
Recall: 0.3660518989335835
F1 Score: 0.4202997107927071
------------------------------------------------

K value: 10
Precision: 0.7322232013269723
Recall: 0.43877250706504817
F1 Score: 0.4504152128147027
------------------------------------------------
```

```
Window size = 30
================================

K value: 3
Precision: 0.6589259796806967
Recall: 0.18326012585720025
F1 Score: 0.185711440158806
------------------------------------------------

K value: 5
Precision: 0.8029148524431514
Recall: 0.2536113309304249
F1 Score: 0.29495337845063624
------------------------------------------------

K value: 7
Precision: 0.7361877116594052
Recall: 0.3062766912785567
F1 Score: 0.34024130747865655
------------------------------------------------

K value: 10
Precision: 0.7010424585896263
Recall: 0.3653202322271295
F1 Score: 0.3828353230100527
------------------------------------------------
```

Those are the results we have got **after we used the evaluation metrics discussed in the paper. As you can see in those results, we have found that the best parameters to use in this algorithm are *k* value should be equal to 10, the *window_size* to be equal to 15, and the *m* multiplier to be equal to 0.5. But to integrate it well with the application and with the below segmentation model, we have decided to assign as the value of the *window_size*.**

2- BiLSTM Model

In this section we show the segmentation results in terms of precision, recall and the F1 score. Our BiLSTM model depends on a threshold for determining a segment boundary. Based on our initial experiments with this model, the threshold could range from 0 till about 0.5, since anything larger would generate very few segments. To determine the optimal thresholds, we performed a grid search when evaluating and testing the model.

Also, we include the results for two tolerance thresholds. A tolerance threshold is how far a predicted segment end time could be from the real segment time to be considered as a correct prediction. Considering that we are working with long lecture videos, we use 30 and 60 seconds. We assume that a tolerance of 60 seconds is very reasonable for our task.

a. Implementation Details

We have implemented this model using PyTorch [41] . To embed sentences, we used a quantized version of the FlagEmbedding model [42]. The quantization was done with the Optimum library from Hugging face [43].

We trained the model for 40 epochs and enable early stopping after completing 10 epochs without improvement. Optimization details are present in Table 7.

Table 7. Optimization Details

BATCH_SIZE	**32**
EPOCHS	40
LEARNING_RATE	0.01
Optimizer	AdamW
Loss Function	Weighted Binary Cross Entropy

We obtain the best model which achieved the a 0.54 F1 score, as shown in Table 8.

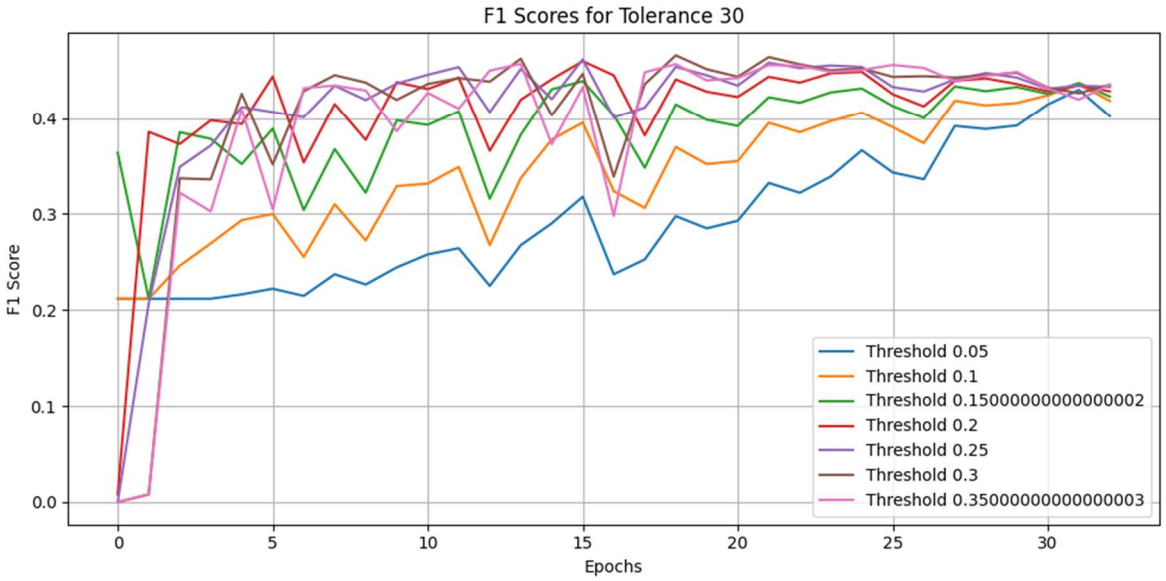
Table 8. Model's F1 Score

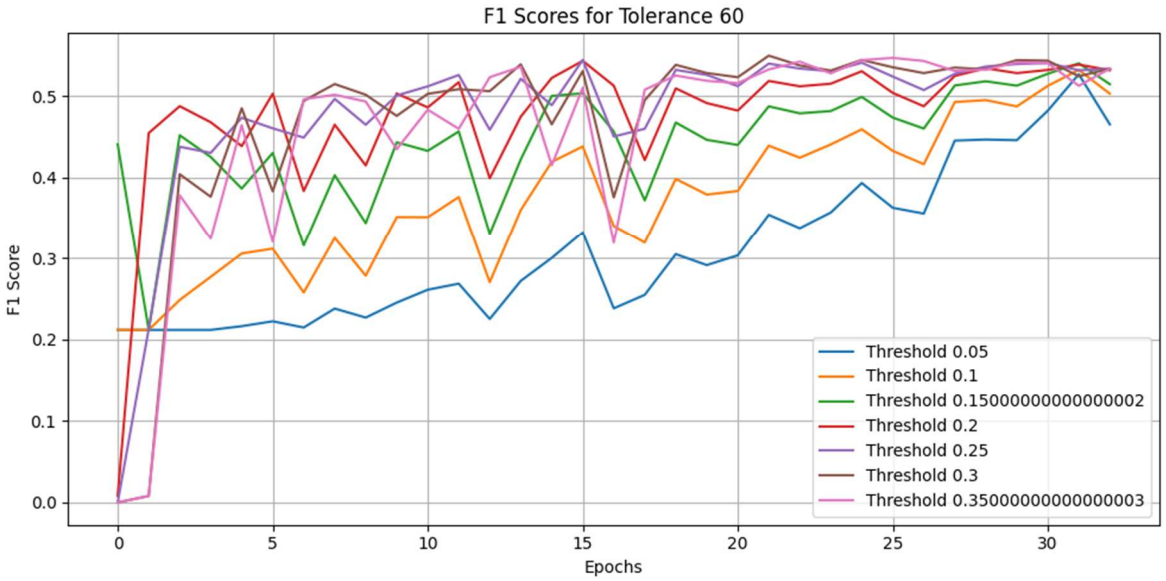
Experiment	Validation loss	F1 score @30s	F1 score @60s
Validation Set	0.169	0.46	**0.54**
Test set	-	0.40	0.47

We have used thresholds in the range of [0.05, 0.35]. The validation loss and F1 scores for both 30- and 60-seconds tolerance are shown in Table 8. The best result, shown in Table x, was obtained with a threshold of 0.3, indicating this is the best threshold in terms of the F1 score.

It should be noted that increasing the threshold will increase the precision and decrease the recall, while decreasing the threshold will have the opposite effect. Therefore, if we want to generate more segments, disregarding how accurate they are, we could decrease the threshold. If we instead wanted more precise segments that occur less frequently, we would increase the threshold.


Training and Validation Losses
Training Loss
Validation Loss
Loss
Epochs


F1 Scores for Tolerance 30
F1 Score
Epochs
Threshold 0.05
Threshold 0.1
Threshold 0.15000000000000002
Threshold 0.2
Threshold 0.25
Threshold 0.3
Threshold 0.35000000000000003


F1 Scores for Tolerance 60
F1 Score
Epochs
Threshold 0.05
Threshold 0.1
Threshold 0.15000000000000002
Threshold 0.2
Threshold 0.25
Threshold 0.3
Threshold 0.35000000000000003

6.1.3 Lecture Summarization

In this section we highlight the latest and most relevant summarization experiments we have performed in this project. These experiments relate to the training and fine-tuning of Abstractive summarization models for lecture video transcript summarization. We note that we have not included all the findings and results, since those can be found in the junior report and other progress reports. We do make some statements and decisions based on these early experiments too.

In this project we wanted to support both summarization of full lecture video from the transcript and the summarization of lecture video segments. The segments approach, as explained above, helps in combating the maximum context length issue. Furthermore, it helps in generating a table of contents for the video after it is summarized. To support these two approaches, we utilized the two datasets mentioned in the Datasets section: AK Lectures and MIT Chapters.

1- **Datasets Splits**

Dataset Name	Training	Validation	Testing
MIT Chapters	11840	1481	1480
AK Lectures	1473	184	185

2- **Implementation Details**

All the experiments presented here were conducted using the Hugging Face Transformers library [44]. This library provides APIs to download models, load datasets, fine-tune models, and perform evaluation among other features. We have fine-tuned the BART models with a batch size of 2, training all the encoder and decoder layers. Additional details in Table 9.

Table 9. Bart Models Training Details

Parameter	Value	Exceptions
Batch size	2	1 for LSG 4 for (MIT Chapters+Title generation)
Gradient Accumulation steps	4	for LSG 4 for (MIT Chapters+Title generation)
Optimizer	*adamw_torch*	-
Scheduler	Linear decay	-
epochs	10	-
Trained Layers	6 Encoder + 6 Decoder layers	LSG: 2 Encoder + 2 Decoder layers

3- **Evaluation Metrics**

We evaluate our models using the Recall-Oriented Underscore Gist Evaluation score, ROUGE. It was introduced in a paper by Lin etc. [46]. This score compares the reference and generated text to find overlapping words, n-grams, or sequences. Using evaluate.load('rouge') will load 4 variants of ROUGE :

- ROUGE-1: The ratio of words that exist in both the generated and reference summary.
- ROUGE-2: The ratio of bi-grams that exist in both the generated and reference summary.
- bi-grams are two words in the same order.
- ROUGE-L: Computes the longest common subsequence that exists in both the generated and reference summaries.
- ROUGE-LSum: Splits the two summaries by newline characters, and measures ROUGE-L on each pair of sentences, then takes the average.

4- **MIT Chapters**

Here we present the results for fine-tuning on MIT chapters summaries. In this experiment, we only trained one model, BART-base, on the full train set of the MIT Chapters.

Here we show the results of fine-tuning the bart-base-cnn checkpoint on the MIT Chapters dataset. The evaluation is done on the validation split. From the wandb dashboard, we could see the training loss over epochs, in Figure 18:

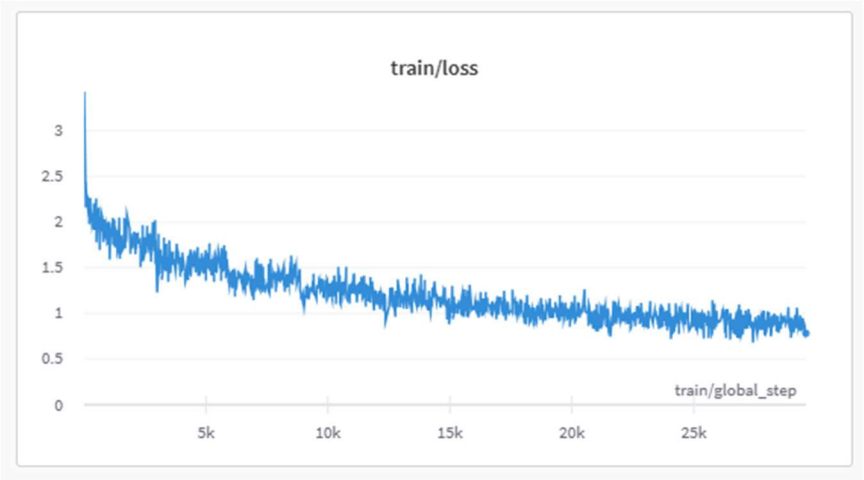


Figure 18. Training loss for bart-base

At each epoch, we evaluate our model by generating summaries and comparing the ROUGE scores to the reference summaries. In Figure 19, we see the rouge scores over the epochs. It increases largely at first but thin stays constant.

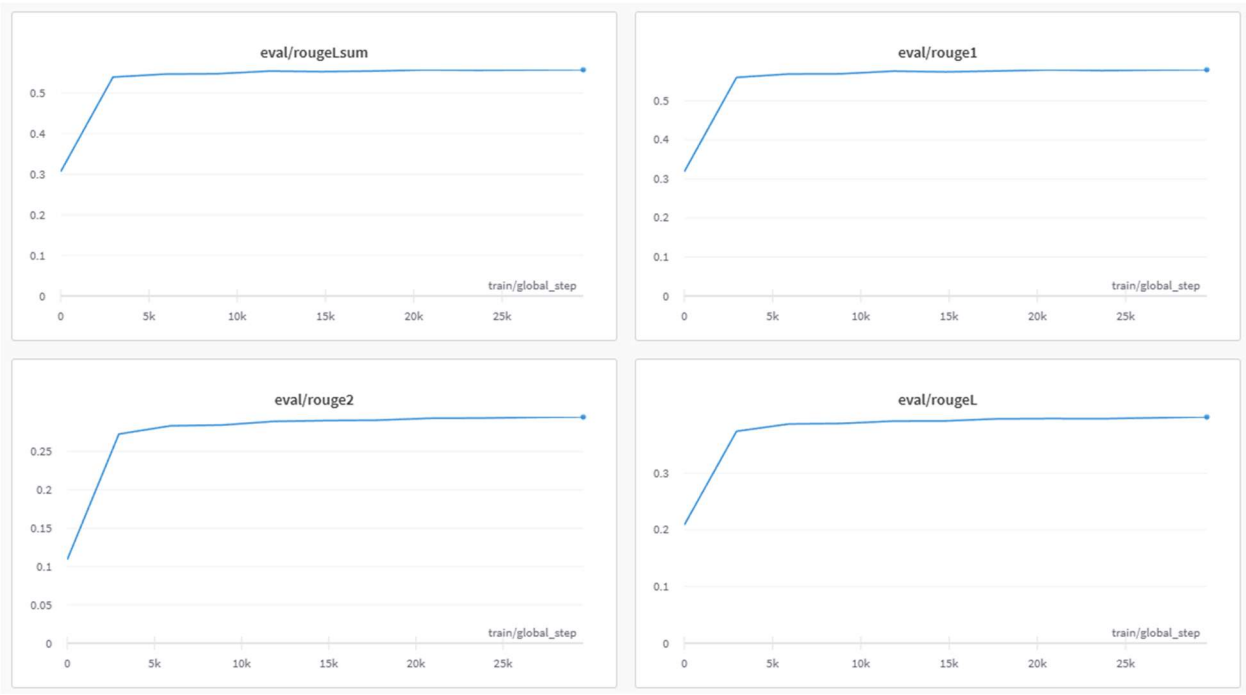


Figure 19. Evaluation results (ROUGE scores) for Bart-base

The initial results are promising. Before fine-tuning, the BART-base-cnn model achieved the following scores on the MIT Chapters validation set:

{'eval_rouge1': 0.32,
'eval_rouge2': 0.11,
'eval_rougeL': 0.21,
'eval_rougeLsum': 0.31}

After fine-tuning for 10-epochs, we reach these scores:

{'eval_rouge1': 0.58,
'eval_rouge2': 0.30,
'eval_rougeL': 0.40,
'eval_rougeLsum': 0.56}

5- **AK Lectures**

We present the results for 3 different types of experiments on the AK Lectures Dataset. The first is fine-tuning BART-base to generate abstractive summaries directly from the original transcript. This experiment limits the input token count to 1024 tokens. The second is utilizing the LSG Attention + BART to extend the limit to 2048 tokens. The third experiment is fine-tuning the model on the extractive summaries instead of the original transcript.

Figure 20. ROUGE Scores on the Validation Set Over Mutliple Epochs. From 5 Experiments That Fine-Tuned The Pre-Trained 'Bart-Base-Cnn' Model.

To generate extractive summaries, we have considered the 3 different techniques mentioned before: LSA [47], Text Rank [48], and BERT [49].

When initializing the LSG Bart model, we loaded a pre-trained checkpoint from Hugging

Face that has been fine-tuned on summarizing scientific articles from the arXiv dataset [45]. This checkpoint greatly aligns without goals. While it was trained on a context length of 4096 tokens, we limit it to 2048 as discussed.

The results of this experiment are shown in the figure below. This figure shows the evaluation results over multiple epochs using the ROGUE score. For convenience, we explain what each name in the plot legend refers to here:

- Bart_base_cnn: Summarize from transcripts directly. Maximum of 1024 tokens.
- Extractive_text_rank/bart_base_cnn: Summarize from TextRank extractive summaries.
- Extractive_lsa/bart_base_cnn: Summarize from LSA extractive summaries.
- Extractive_bert/bart_base_cnn: Summarize from BERT extractive summaries.
- Lsg_bart_base_arxiv: Summarize from transcripts directly. Maximum of 2048 tokens.

6- **Discussion**

Surprisingly, we notice that the experiment of summarizing from the transcript directly achieves the best result across all scores. This is surprising because the model is only seeing the first 1000 of tokens. The TextRank and LSA extractive summary techniques come next and are almost equal. The LSG method scores very close to both methods, while the BERT extractive method has the lowest score.

We could explain the difference in scores between the base method and other extractive techniques with the assumption that extractive methods can remove important information, there these are omitted from the final abstractive summary which reduces the score. This is corroborated by the fact that these methods have not been fine-tuned for lectures, and we have no method to measure the quality of the summaries they extract.

The LSG method in this case has been trained with 4 layers only (2 from encoder and 2 from decoder), which is an exception to all other models. This can interpret the relatively lower score, even though we are seeing more input tokens.

7- **AK Lectures + MIT Chapters**

In this section, we present the best results we obtained for summarizing full lecture videos from transcripts. The findings in this experiment all build up on our investigations and initial experiments in both the junior and senior report. After fine-tuning BART-base on the MIT Chapters results and achieving a reasonably impressive ROUGE1 score of 0.58, we decided we could make use of the Transfer Learning [50] technique once more. In this case, we wanted to benefit from the knowledge the model gained in summarizing lecture segments (chapters) and use it in summarizing full lecture videos.

Therefore, we have re-done all experiments on the AK Lectures dataset, under the same setup, but with the difference that we initialized the BART-base model from the weights of the best model that was fine-tuned on the MIT chapters dataset. One problem we faced is that the LSG model was not fine-tuned on MIT chapters, so we re-did the first experiment but using the lsg-bart-base-arxiv model, limited to 2048 tokens, and then used this model in this experiment.

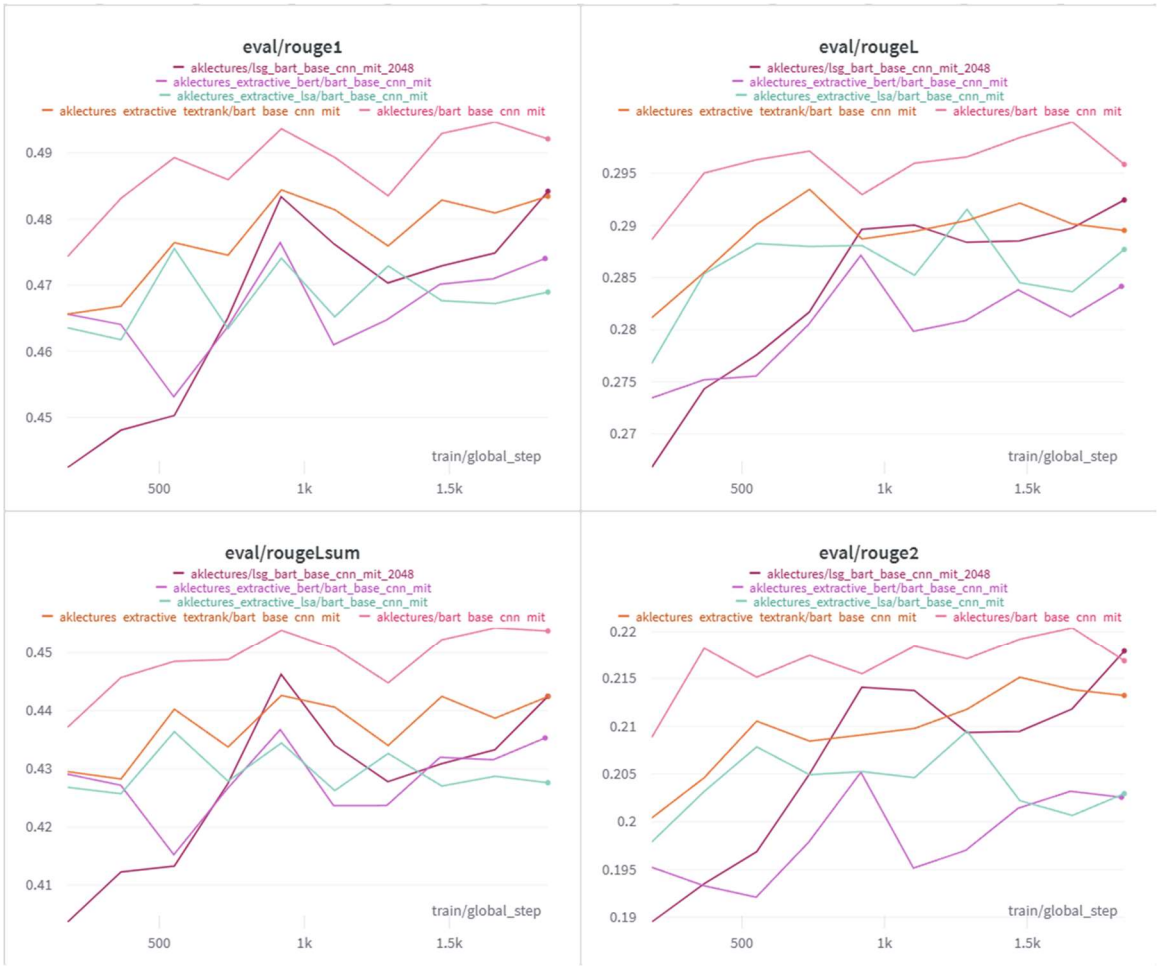


Figure 21. Great Improvement in the Scores

As can be seen in Figure 21, we see a great improvement in the scores over the BART-base model that was fine-tuned on CNN alone. The worts Rouge1 score jumped from 0.45 to 0.47, and the best score increased from 0.46 to 0.49. In this experiment, we also notice that the LSG model still falls short of the best model, which could be because the original lsg-bart-base-cnn-mit model was fine-tuned with partially frozen layers. It should be noted that in this specific experiment, LSG was fine-tuned with no frozen layers.

8- **MIT Chapters + Titles**

After we finished implementing and testing the segmentation algorithms, we decided we wanted to add support for title generation from each transcript segment. As discussed earlier, we tried multiple approaches that failed to provide a good result. The last approach we considered was integrating this task into the summarization task and fine-tuning the BART-base-cnn-mit model again to also generate titles in addition to segment summaries. In this section we show the results of this experiment.

To set up the data for this experiment, we simply prepend each segment summary with the title in the first line. This experiment was performed on a Google Collaboratory environment, with a V100 GPU. Therefore, we were able to increase the batch size up to 4, with a gradient accumulation step of 4 as well, making the effective batch size equal to 16. Also, we only fine-tune for about 3 epochs since the validation loss stopped decreasing after that.

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum
1	0.939600	0.732972	0.642526	0.396760	0.483883	0.624612
2	0.851700	0.729540	0.642692	0.395220	0.482111	0.625303
3	0.782400	0.723357	0.646332	0.397682	0.485561	0.627981

Figure 22. Results for fine-tuning BART-base, previously fine-tuned for segment summarization, to also generate titles for each segment.

```
Reference Title:
 Difference Equations

Reference Summary:
 The lecture discusses difference equations and their similarities to differential equations. It
explains that a linear constant coefficient difference equation is a linear combination of delayed
versions of the output equal to a linear combination of delayed versions of the input. The lecture
also mentions that a difference equation is not a complete specification of the system because a
homogeneous solution can be added to the response. The homogeneous solution is a linear combination
of exponentials. The lecture then explains how to determine the coefficients of the solution through
auxiliary conditions, such as initial conditions or boundary conditions. It emphasizes the
importance of the system being linear and causal, and how the auxiliary conditions need to be
consistent with initial rest. The lecture concludes by stating that difference equations, assuming
causality, provide an explicit input-output relationship for the system. It also explains how to
compute the output for the next time instant and how to get the equation started through appropriate
initial conditions.

Generated Title: Difference Equations

Generated Summary: Difference Equations
In this section of the lecture, the speaker discusses difference equations and their application in
differential equations. They explain that a linear constant coefficient difference equation is a linear
combination of delayed versions of the output equal to a linear combination of delayed
versions of the input. The speaker emphasizes that difference equations are not a complete specification of the
system, as any solution that satisfies the homogeneous equation and the sum of those solutions
will also satisfy the original difference equation. The homogeneous solution is of the form of a
linear combination of exponentials, where the amplitude factor is undetermined and needs to be determined through
appropriate initial conditions or boundary conditions. The general form for the solution to the difference
equation is a sum of exponential coefficients and N undetermined coefficients, and N auxiliary conditions
are required to determine these coefficients. For linearity, the response must be equal to 0, and the
audience must be causal and consistent with initial rest, meaning that if the input is 0
prior to a certain time, the output must also be 0 prior to the same time. The
speaker also mentions that a difference equation assumes an explicit input-output relationship
for the system, and rearranges the equation to express the output in terms of prior values of
the input. Once the system is started, it can be computed for the next time instant using appropriate initial
conditions, boundary conditions, and initial rest.
```

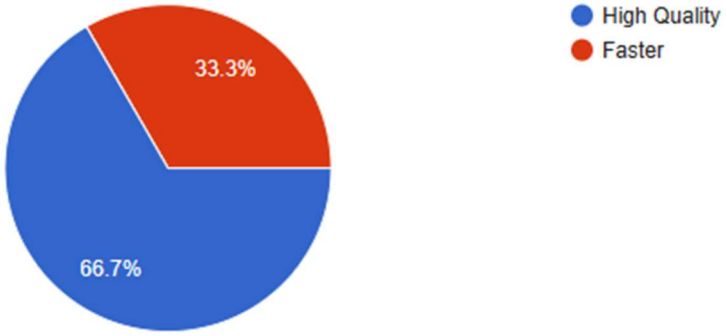
Figure 23 Example of a reference title and summary, and the corresponding generated title and summary.

6.1.4 Beta Testing

In this phase, we have brought different types of users to test LASER. We first allowed them to try the application and at the end to fill out a google form by answering the questions that were posted there.

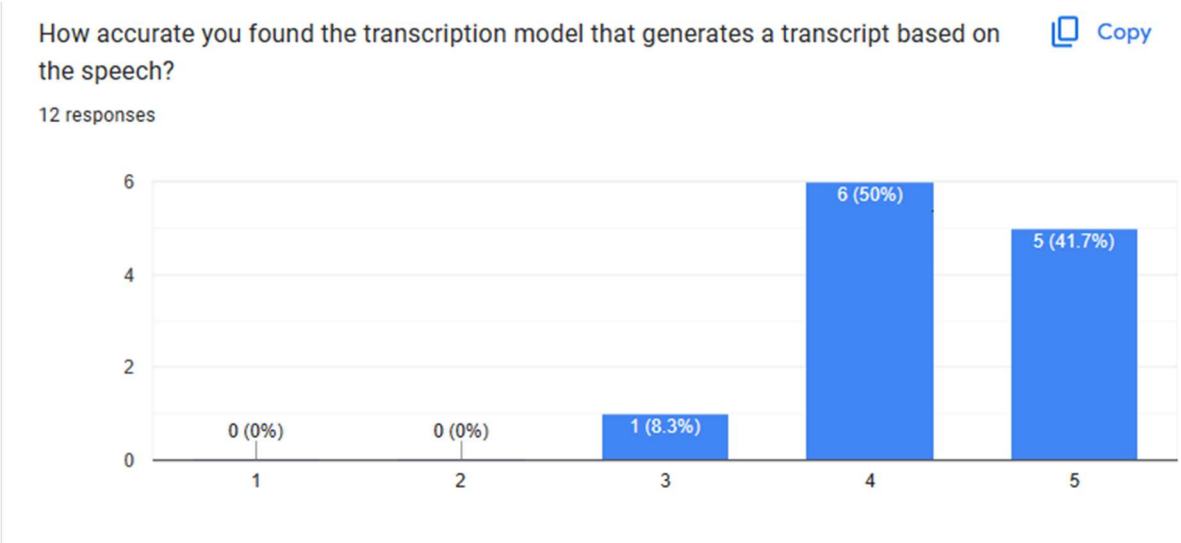
We will show the results of this experiment by demonstrating the results that has been in the google form.

1- Types of Segmentation used:



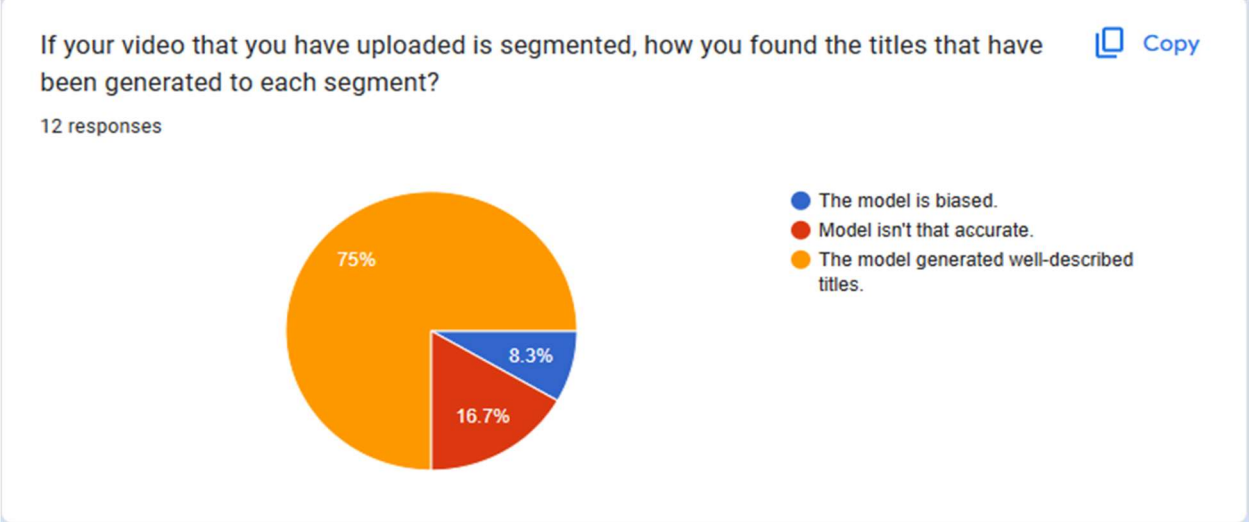
The users in this testing phase **have used the high-quality segmentation more.**

2- Accuracy of the transcription model:



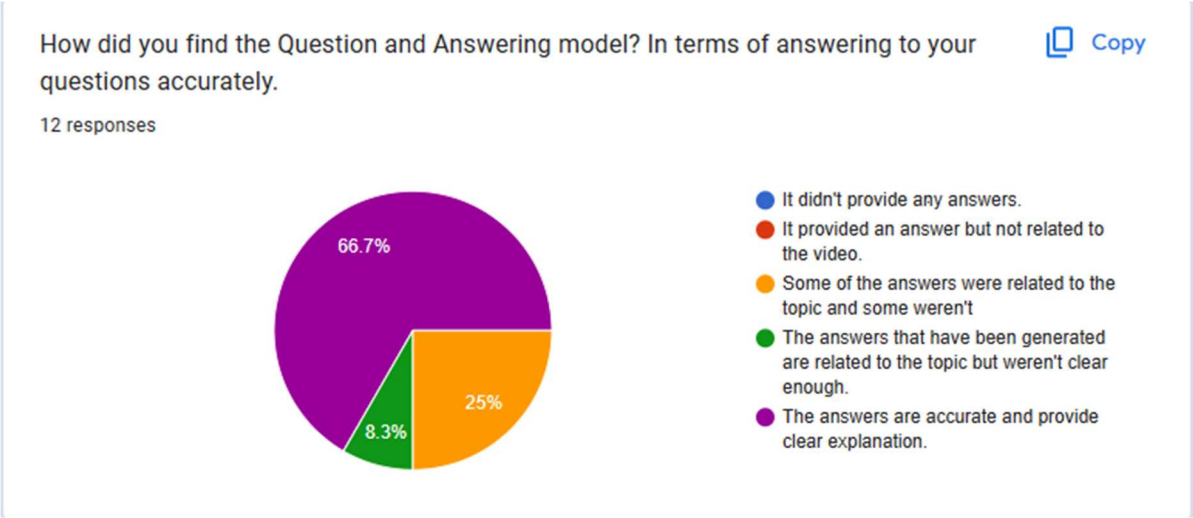
The results shown **how highly accurate the transcripts that have been generated.**

3- Accuracy of the Segmentation techniques and the generated titles:



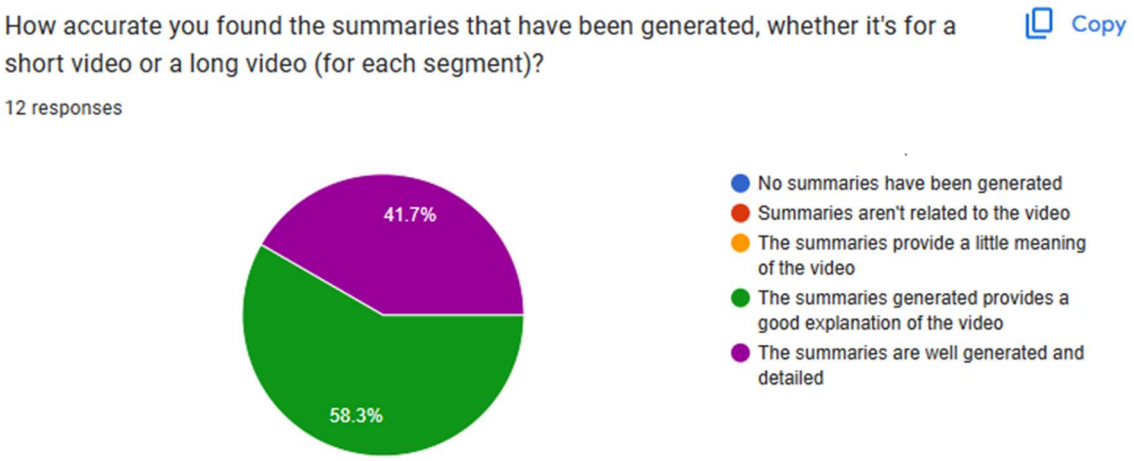
The results show how well-described the titles are and the how much accurate the segments are. Despite sometimes the model generate a redundant title, most of the testers got well-described titles that have meaning of each segment.

4- Accuracy of the Question and Answering model:



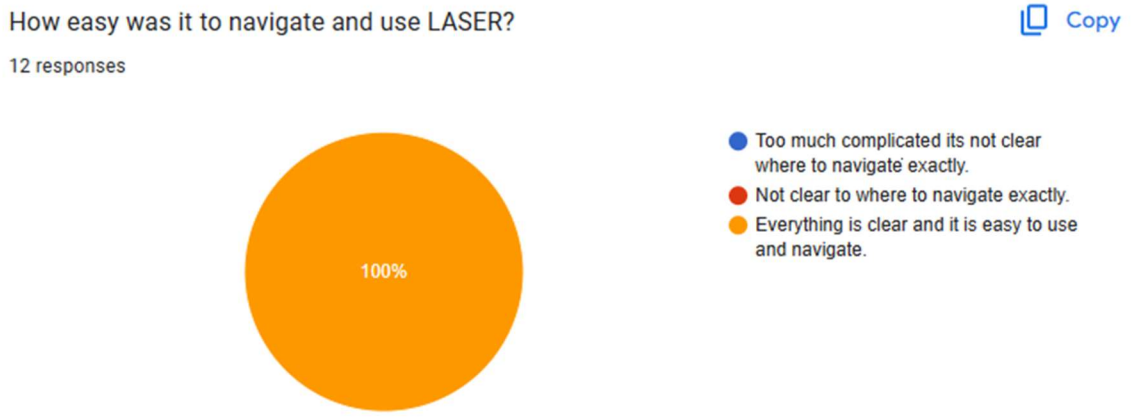
Despite sometimes the generated answers were out of topic based on the testers question, the answers that the model generates are well defined and provides a clear explanation that will help users understand their recordings more.

5- Accuracy of the Summarizations models:



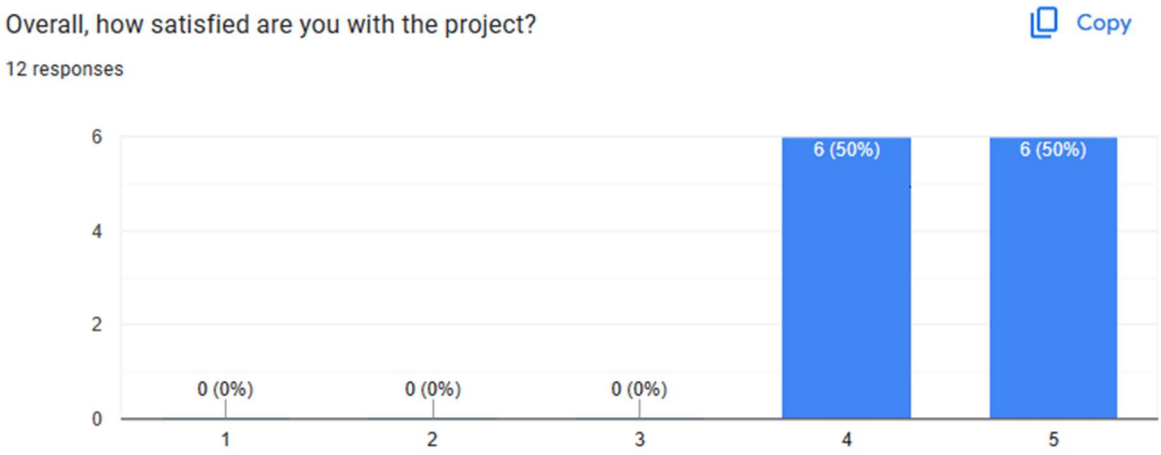
The above results illustrate how our summarization models provide a clear context and detailed explanation of the recordings.

6- Using LASER:



This diagram proves how LASER is easy to use and navigate for users.

7- Satisfactions of the testers:



This diagram illustrates how happy and satisfied the testers were with our application.

During the two-day testing phase, we checked how well our application performs compared to others in its field. The results highlighted that our solution is strong and has a competitive edge. We've showcased positive reviews from users on our landing page, giving a quick look at what users think. We also conducted detailed interviews with testers who shared how LASER is useful and pinpointed the groups that would benefit the most from its features. All these careful testing methods and user feedback confirm that our application works effectively.

CHAPTER 7: Conclusion and Future Goals

In the journey of developing the LASER (Learning from Automatically Summarized and Segmented Educational Recordings) web application, our team has made significant strides in the realms of transcription, segmentation, summarization, and question-answering for educational videos. This final report encapsulates our comprehensive efforts, challenges faced, and the innovative solutions devised to enhance the educational experience for users.

7.1 Achievements and Contributions

1- Datasets Exploration:

Despite initial challenges in finding suitable datasets for our tasks, we successfully extended the AK Lectures dataset, creating the MIT Chapters dataset. This expansion enabled us to address specific needs and challenges in segmentation and summarization.

2- Technical Background:

The implementation of the LASER web application involved a detailed exploration of Automatic Speech Recognition (ASR) solutions, leveraging Faster Whisper and Flash Attention v2 for efficiency gains. The integration of diverse models, including BART for summarization and BiLSTM for segmentation, marked crucial advancements.

3- Innovative Summarization Approaches:

Our exploration into abstractive summarization models led us to fine-tune BART models on datasets like MIT Chapters and AK Lectures. Creative solutions, such as generating titles and summaries simultaneously, significantly improved ROUGE scores, demonstrating the versatility of our approach.

4- User-Centric Features:

LASER was designed with user convenience in mind. Features like searching, question-answering, and the ability to summarize both full lectures and segmented chapters empower users to navigate, comprehend, and revisit their educational content seamlessly.

5- Three-Tier Application Architecture:

The LASER web application's robust architecture, with Flask on the server side and React on the client side, ensures a smooth and responsive user experience. The incorporation of AI models for transcription, segmentation, and summarization enriches the educational content.

7.2 Extended Future Goals

In addition to our core development objectives, the LASER project envisions broader impact and collaboration, solidifying its position as an open-source, community-driven tool:

1- Open-Source Community Building:

Foster an active open-source community around LASER, encouraging developers, educators, and researchers to contribute, enhance functionalities, and address emerging challenges collectively. Establishing forums, documentation, and collaborative platforms will be instrumental in this endeavor.

2- Public Release:

Make LASER accessible to the public by releasing it as an open-source tool. This move will democratize access to advanced educational video processing tools, enabling a wider audience to benefit from LASER's capabilities.

3- Research Dissemination:

Share insights, methodologies, and outcomes with the academic community by preparing and submitting conference papers. This will contribute to the collective knowledge in the fields of natural language processing, educational technology, and AI-driven tools for content analysis.

4- Expanded Training Datasets:

Enhance the robustness of our summarization models by incorporating additional training data. Consider scraping educational content from platforms like Khan Academy to create a more diverse and comprehensive dataset, improving the models' adaptability to varied educational styles and content structures.

5- Human-Annotated Segmentation:

Improve the segmentation AI model by training it on human-annotated segments from YouTube. This approach ensures a more nuanced understanding of educational video structures, considering the expertise of human annotators in identifying meaningful segments.

6- Question Answering Model Enhancement:

Further strengthen our Question Answering model by training it on a more extensive dataset. Explore avenues to acquire diverse datasets, potentially collaborating with educational institutions or tapping into publicly available question-answer datasets.

LASER's journey goes beyond individual technological advancements; it aspires to create a collaborative ecosystem that empowers users, researchers, and developers alike. By embracing open-source principles, engaging with the wider community, and continuously innovating, LASER aims to redefine the landscape of educational video processing tools.

CHAPTER 8: References

[1] "Fireflies.ai | AI notetaker to transcribe, summarize, analyze meetings," Fireflies. https://fireflies.ai/

[2] Video Conferencing & meeting Software | Powered by AI. (n.d.). Vowel. https://www.vowel.com/

[3] Eightify, "YouTube AI Video Summaries with Eightify AI ChatGPT," Eightify Youtube Summaries. https://eightify.app/

[4] "summarize.tech: AI-powered video summaries." https://www.summarize.tech/

[5] "Jasper | AI copilot for enterprise marketing teams." https://www.jasper.ai/

[6] "Text Summarizer | QuillBot AI." https://quillbot.com/summarize

[7] Tommy, "Frase - Best SEO Content Optimization Tool & AI Writer," Frase, Nov. 28, 2023. https://www.frase.io/

[8] "Summarize website and summarize text," Copyright Smodin LLC 2023. https://smodin.io/text-summarizer

[9] "Website Summary AI | Ask AI anything about your website," Website Summary AI. https://www.websitesummaryai.com/?ref=taaft&utm_source=taaft&utm_medium=referral

[10] These 3 charts show the global growth in online learning. (2023, May 1). World Economic Forum. https://www.weforum.org/agenda/2022/01/online-learning-courses-reskill-skills-gap/

[11] 39 Interactive Learning Statistics: 2023 Data, Trends & Predictions. (2023, October 31). Research.com. https://research.com/education/interactive-learning-statistics

[12] "66 elearning Statistics: 2023 Data, analysis & Predictions," Research.com, Nov. 10, 2023. [Online]. Available: https://research.com/education/elearning-statistics

[13] Gupta, D., & Sharma, A. (2023). A comprehensive study of automatic video summarization techniques. Artificial Intelligence Review, 56(10), 11473–11633. https://doi.org/10.1007/s10462-023-10429-z

[14] F. Sun and X. Tian, "Lecture Video Automatic Summarization System based on DBNET and Kalman Filtering," Mathematical Problems in Engineering, vol. 2022, pp. 1–10, Aug. 2022, doi: 10.1155/2022/5303503

[15] T. Tuna and R. Verma, “Topic based segmentation of classroom videos,” Uh, May 2016, [Online]. Available: https://www.academia.edu/25313906/Topic_Based_Segmentation_of_Classroom_Videos

[16] Lgordon. (2022, April 1). Introducing the Coursera Impact Report 2021. Coursera Blog. https://blog.coursera.org/coursera-impact-report-2021/

[17] Özdemir, S. (n.d.). The Effect of Summarization Strategies Teaching on strategy usage and narrative text summarization success. https://eric.ed.gov/?id=EJ1192722

[18] Admin, “The benefits of transcribing educational research,” Jul. 21, 2023. https://transcriptionus.com/blog/the-benefits-of-transcribing-educational-research/

[19] Educ5104g, P. I. (2020, April 10). Segmenting principle. Pressbooks. https://pressbooks.pub/elearning2020/chapter/segmenting-principle/

[20] Shanmugavelu, G., Ariffin, K., Vadivelu, M., Mahayudin, Z., & Sundaram, M. a. R. K. (2020). Questioning techniques and teachers’ role in the classroom. Shanlax International Journal of Education, 8(4), 45–49. https://doi.org/10.34293/education.v8i4.3260

[21] Heick, T. (2022, January 28). An updated guide to questioning in the classroom. TeachThought. https://www.teachthought.com/critical-thinking/questioning-guide/

[22] The challenges and opportunities of Artificial Intelligence in education. (2023, April 20). UNESCO. https://www.unesco.org/en/articles/challenges-and-opportunities-artificial-intelligence-education

[23] Aiforgoodstg, “How can artificial intelligence improve education?,” AI For Good, Sep. 02, 2021. https://aiforgood.itu.int/how-can-artificial-intelligence-improve-education/

[24] “28 Video training Statistics: 2023 Data, Trends & Predictions,” Research.com, Oct. 31, 2023. [Online]. Available: https://research.com/education/video-training-statistics

[25] R. Knott, “Video Statistics, Habits, and trends you need to know,” The TechSmith Blog, Dec. 14, 2022. https://www.techsmith.com/blog/video-statistics/

[26] “How AI can accelerate students’ holistic development and make teaching more fulfilling,” World Economic Forum, Oct. 06, 2023. https://www.weforum.org/agenda/2023/05/ai-accelerate-students-holistic-development-teaching-fulfilling/

[27] A. Pandey, “Benefits of Video-Based Learning,” eLearning, Apr. 28, 2022. https://elearning.adobe.com/2022/04/benefits-of-video-based-learning/

[28] S. Hart, "3 Key benefits of Video-Based Learning You should not ignore," eLearning Industry, May 12, 2021. https://elearningindustry.com/key-benefits-of-video-based-learning

[29] "Artificial intelligence in education," UNESCO, Nov. 29, 2023. https://www.unesco.org/en/digital-education/artificial-intelligence

[30] Educating in a world of artificial intelligence. (2023, February 9). Harvard Graduate School of Education. https://www.gse.harvard.edu/ideas/edcast/23/02/educating-world-artificial-intelligence

[31] SYSTRAN/faster-whisper: Faster Whisper transcription with CTranslate2. (n.d.). GitHub. https://github.com/SYSTRAN/faster-whisper

[32] Pipelines. (n.d.). https://huggingface.co/docs/transformers/main_classes/pipelines

[33] "Dao-AILab/flash-attention: Fast and memory-efficient exact attention," GitHub. https://github.com/Dao-AILab/flash-attention

[34] "FlagOpen/FlagEmbedding: Dense Retrieval and Retrieval-Augmented LLMs," GitHub. https://github.com/FlagOpen/FlagEmbedding

[35] "sentence-transformers/msmarco-distilbert-base-tas-b · Hugging Face," Apr. 05, 2001. https://huggingface.co/sentence-transformers/msmarco-distilbert-base-tas-b

[36] TheBloke/Llama-2-7B-Chat-GGUF · Hugging face. (n.d.). https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGUF

[37] R. Savaram, "Introduction to ReactJS," Mindmajix, Apr. 03, 2023. https://mindmajix.com/introduction-to-react-js

[38] Babu, R. (2022, March 12). Building a scalable and modular architecture for React-TS applications. Medium. https://levelup.gitconnected.com/building-a-scalable-and-modular-architecture-for-react-ts-applications-e1d917250e04

[39] P. Tiwari, "CORS error in react," Scaler Topics, May 03, 2023. https://www.scaler.com/topics/react/cors-error-in-react/

[40] "BentoML: Build, Ship, Scale AI Applications." https://www.bentoml.com/

[41] *PyTorch*. (n.d.). PyTorch. https://pytorch.org/

[42] *"FlagOpen/FlagEmbedding: Dense Retrieval and Retrieval-Augmented LLMs,"* *GitHub. https://github.com/FlagOpen/FlagEmbedding*

[43] *"🤗 Optimum." https://huggingface.co/docs/optimum/index*

[44] *" 🤗 Transformers."* https://huggingface.co/docs/transformers/index

[45] *"ccdv/lsg-bart-base-4096-arxiv · Hugging Face," Apr. 05, 2001.* https://huggingface.co/ccdv/lsg-bart-base-4096-arxiv

[46] *Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. Text Summarization Branches Out, 74–81.* https://aclanthology.org/W04-1013

[47] *Kireyev, K. (n.d.). Using Latent Semantic Analysis for Extractive Summarization*

[48] *Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing Order into Text. In D. Lin & D. Wu (Eds.), Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (pp. 404–411). Association for Computational Linguistics.* https://aclanthology.org/W04-3252

[49] *Liu, Y. (2019). Fine-tune BERT for Extractive Summarization (arXiv:1903.10318). arXiv.* http://arxiv.org/abs/1903.10318

[49] *Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A Comprehensive Survey on Transfer Learning. Proceedings of the IEEE, 109(1), 43–76.* https://doi.org/10.1109/JPROC.2020.3004555

[50] *Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020). PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. Proceedings of the 37th International Conference on Machine Learning, 11328–11339.* https://proceedings.mlr.press/v119/zhang20ae.html

[51] *Ainize/bart-base-cnn · Hugging Face*. (2022, November 18). https://huggingface.co/ainize/bart-base-cnn

[52] Cajueiro, D. O., Nery, A. G., Tavares, I., De Melo, M. K., Reis, S. A. dos, Weigang, L.,

& Celestino, V. R. R. (2023). *A comprehensive review of automatic text summarization techniques: Method, data, evaluation and coding* (arXiv:2301.03403). arXiv. http://arxiv.org/abs/2301.03403

[53] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension* (arXiv:1910.13461). arXiv. http://arxiv.org/abs/1910.13461

[54] Liu, Y., & Lapata, M. (2019). *Text Summarization with Pretrained Encoders* (arXiv:1908.08345). arXiv. http://arxiv.org/abs/1908.08345

[55] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, *21*(1), 140:5485-140:5551

[56] "AK Lectures." https://aklectures.com/

[57] "MIT OpenCourseWare," *YouTube*. https://www.youtube.com/@mitocw