

A Major Project Report On  
**SPEECH TO SIGN LANGUAGE CONVERSION USING 3D MODEL**  
Submitted in partial fulfillment of the requirements for the award of the

**Bachelor of Technology**

In

**Department of Computer Science and Engineering**

By

**Dodda Abhiram**

**21241A0511**

Under the Esteemed guidance of

**Tabitha Indupalli**

**Assistant Professor of CSE**



**Department of Computer Science and Engineering**  
**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY**  
**(Autonomous)**  
**Bachupally, Kukatpally, Hyderabad, Telangana, India, 500090**  
**2024-2025**



**GOKARAJU RANGARAJU**  
**INSTITUTE OF ENGINEERING AND TECHNOLOGY**  
**(Autonomous)**

**CERTIFICATE**

This is to certify that the major project work entitled “**Speech to Sign Language Conversion using 3D Model**” is submitted by **DODDA ABHIRAM (21241A0511)**, in partial fulfillment of the award of a degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2024-2025**.

GUIDE  
**TABITHA INDUPALLI**  
**Assistant Professor**

HEAD OF THE DEPARTMENT  
**Dr. B. SANKARA BABU**  
**Professor**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our guide **Tabitha Indupalli, Assistant Professor**, Department of CSE for her support in the completion of our project work successfully and for all the time to time guidance provided. The process made us to learn a lot practically and we are very thankful to the final year Project Coordinator **Dr. N. Krishna Chythanya**, Asst. Prof., CSE and our class project coordinator **Ms. Asha Reddy**, Asst. Prof., CSE for their unwavering support in providing schedules, formats, rubrics and arranging the seminars at regular intervals. Our Sincere thanks to Project Review Committee member **Dr. G. R. Sakthi Dharan, Professor**, CSE, for their in-depth analysis during evaluations of seminars that helped us in improvising the quality of our work. We wish to express our honest and sincere thanks to **Dr. B. Sankara Babu, HOD**, Department of CSE, to our Principal **Dr. J. Praveen** and to our Director **Dr. Jandhyala N Murthy** for providing all the facilities required to complete our major project. We would like to thank all our faculty and friends for their help and constructive criticism during the completion of this phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

**Dodda Abhiram**

**21241A0511**

## **DECLARATION**

We hereby declare that the major project entitled “**SPEECH TO SIGN LANGUAGE CONVERSION USING 3D MODEL**” is the work done during the period from **2024-2025** and is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous)**. The results embodied in this phase-I of project have not been submitted to any other University or Institution for the award of any degree or diploma.

**Dodda Abhiram**

**21241A0511**

	<b>Table of Contents</b>	
<b>Chapter</b>	<b>TITLE</b>	<b>Page No.</b>
	Abstract	IX
1	Introduction	1
2	Literature Survey	3
	2.1 Introduction	3
	2.2 Related Work	4
	2.3 Gaps Identified	8
	2.4 Problem Statement	9
	2.5 Proposed Solution	9
	2.6 Summary	10
3	System Requirements Specification	11
	3.1 Introduction	11
	3.1.1 Purpose	11
	3.1.2 Scope	11
	3.1.3 Definitions, Acronyms and Abbreviations	13
	3.1.4 References	13
	3.1.5 Overview	13
	3.2 General Description	14
	3.2.1 Product Perspective	14
	3.2.2 Product Functions	14
	3.2.3 User Characteristics	15
	3.2.4 General Constraints	16
	3.2.5 Assumptions and Dependencies	17
	3.3 Specific Requirements	17
	3.3.1 Functional Requirements	17
	3.3.2 Non- Functional Requirements	18
	3.3.3 User Interface Requirements	18
	3.3.4 Hardware Requirements	19
	3.4 Architecture and UML Diagrams	21
	3.4.1 Architecture Diagram	21
	3.4.2 Use Case Diagram	22
	3.4.3 Class Diagram	23
	3.4.4 Flowchart	25
	3.5 Summary	25
4	Methodology	26
	4.1 Modules	26
	4.2 Methodology Proposed For Solution Implementation.	30
	4.3 Implementation	32
	4.4 Summary	47

5	Results	48
	5.1 Unit Testing	48
	5.2 Results	49
6	Conclusion and Future Scope	50
	6.1 Conclusion	50
	6.2 Future Scope	50
7	References	51

	<b>LIST OF FIGURES</b>	
<b>Fig No</b>	<b>Fig Title</b>	<b>Page No</b>
1	<b>Architecture Diagram</b>	21
2	<b>Use case Diagram</b>	22
3	<b>Class Diagram</b>	23
4	<b>Activity Diagram</b>	24
5	<b>Flow Chart</b>	25
6	<b>Architecture of Whisper Model</b>	26
7	<b>Architecture of RNN used for denoising</b>	27
8	<b>Architecture of BERT</b>	28
9	<b>Transformer architecture</b>	29
10	<b>Architecture of Encoder used in BERT</b>	30
11	<b>BERT for keyword extraction</b>	32
12	<b>User Login Page</b>	48
13	<b>Invalid User Login</b>	48
14	<b>User Registration Form</b>	48
15	<b>Application Dashboard</b>	49
16	<b>Converting Speech to Video</b>	49
17	<b>Final Video Output</b>	49

# **ABSTRACT**

## **SPEECH TO SIGN LANGUAGE CONVERSION USING 3D MODEL**

The "Speech to Sign Language Conversion using 3D Model" project would help bridge communication gaps that there are among heard and the hearing-impaired community. It uses advanced NLP techniques and 3D modeling to make sure that spoken speech was captured and converted into text first and then to later process through natural language processing to bring forth meaningful keywords out. These keywords will then be mapped to their relating signs, which will later be demonstrated by a prepared 3D model capable of performing sign language gestures. It will automatically translate the process and offer an easy convertibility of a spoken language into sign language in real-time. This innovation can easily make headlines in bringing accessibility to hearing-impaired individuals with wider inclusivity of communication.

This technology has a variety of applications across environments such as interpersonal dialogue, educational environments, and customer service solutions. As a dynamic and mechanized solution over static approaches using human interpreters or static software, this system reduces reliance on external aid and further supports independence. This technology will significantly increase direct communication as well as further support the building of a more inclusive community, as its development continues to mature and its integration into widely adopted devices further enhances.



# 1. Introduction

"Speech to Sign Language Conversion using 3D Model" project uses a variety of advanced technologies that bridge the gap for communication from the hearing population to the deaf population. This system has been designed in such a way that it can interpret spoken language instantly and convert it into sign language by using speech recognition, natural language processing, and three-dimensional modeling. They synergistically work together to ensure that a sound, effective solution is deployed in automatic translation for communication.

The first step in the process is speech recognition technology, which captures oral input and converts it into written words. The modern speech recognition system uses a deep learning-based algorithm. These are created and designed on huge quantities of data that will be allowed to have very high accuracy while working with different accents, speech characteristics, and noises in the background. After analyzing the spoken word and encoding it in text, the system creates a foundation to process and translate the text into sign language.

After speech is transcribed into text, then NLP comes in handy in the process. The set of NLP techniques helps analyze the content of the text for key words and to get meaning from said communication. Techniques like tokenization, part-of-speech tagging and semantic analysis ensure that only important information is sent. Since sign language comprises rather delicate expressions most of which are impossible to translate literally, it requires a broad contextual understanding to ensure the sign language gestures produced reflect the intentions of the speaker.

The user-defined gestures are represented to the world through the use of a system called 3D modeling technology. This involves a pre-developed 3D model of a human avatar programmed to display accurate and fluid sign language gestures.

The generation of animations that closely match hand movements, facial expressions, and body languages is an integral part of the general effort to imitate sign communication. 3D modeling, therefore, allows for great flexibility with regard to modifying and expanding the system's gesture library with new signs being developed or languages being added. The realistic, interactive nature of 3D models intensifies user experience, thereby making the system both interesting and accessible to everybody: both listeners and non-listeners.

All these technologies together form an integrated pipeline that processes the spoken input, interprets it, and returns precise representations of what is said in sign language just at the moment it has been said. This highly innovative approach solves many of the limitations of the current solutions based on human interpreter or fixed resources such as flashcards. As this translation process is automated, it pushes towards scalable applicability in numerous aspects ranging from everyday conversations to educational environments and service frameworks. This project holds vast potential for reshaping accessibility in the deaf and hard-of-hearing community, thus making society much more inclusive. The system, using Speech Recognition, NLP, and 3D modeling, presents a superior, real-time communication tool with a guarantee of complete participation by people with hearing impairments in actions and conversations. This marks a large stride toward the creation of automated solutions intended to increase access. It emphasizes the kinds of innovation required to foster a society that values inclusion and diversity.

## 2. Literature Survey

### 2.1 Introduction

Communication is an integral part of human interaction, yet millions of people throughout the world suffer major obstacles in listening. For several individuals within the deaf community, sign language is their primary mode of communication, but a lack of sign language proficiency among the majority of people frequently disrupts effective communication. This project seeks to bridge this gap by designing a real-time system for the speech-to-sign language translation. The solution is designed to empower those with a hearing impairment in that they are enabled to communicate freely with the talking population, and this indeed creates an atmosphere of inclusiveness and accessibility.

This project springs from the motivation of combining the dream of an inclusive society by bridging the communication divide with the route of integrating technology. The emergence of artificial intelligence now makes it feasible to design intelligent systems that could interpret spoken language and translate it to corresponding sign language gestures. Thus, the project focuses on building functional, language-independent, and scalable systems that suit regional variations of sign language.

The project integrates cutting-edge technology across various fields. Speech-to-text processing is done by OpenAI's Whisper model, which proves to be robust in processing various accents, speech speeds, and background noise. Whisper's capability for multilingual support offers inclusivity for non-native speakers. Natural Language Processing tools like BERT and T5 are used for processing text. These tools assist in keyword extraction, syntax reduction, and grammar reformulation to make the text suitable for the sign language pattern. Pre-existing sign language databases are used to animate gestures, while generating new gestures as required utilizes tools like Blender, thus offering an inclusive and dynamic system. The gestures are translated using a virtual avatar, which offers a crisp and interactive representation of the translated speech.

The system has vast potential for use in different fields, ranging from education to healthcare, customer service, social life, and the workplace. In the education field, it can offer vast benefits to hearing-impaired students through the provision of the capability to access lectures at the same time as real-time interpretation. Organizations can use the system to help hearing-impaired clients, while workplaces can use it to make meetings and team activities more

inclusive. The system can also enhance social relationships through the provision of instant communication between hearing-impaired people and the general speaking population.

By combining robust speech recognition, sophisticated NLP techniques, and visually engaging sign language animations, this project aims to create a transformative tool. It is a step forward in improving accessibility and fostering inclusivity, ensuring that hearing-impaired individuals can actively participate in all aspects of society.

## **2.2 Related Work**

The researchers of [1] proposed a model that aimed to leverage BERT for a text classification task. The main objective is to accurately predict the label associated with a given input text the technique used is the transformer architecture masked language modelling next sentence prediction the architecture used is Multi-layer bidirectional transformer encoder. The evaluation metrics used are GLUE (General Language Understanding Evaluation) and has 80.5% score and MultiNLI of 86.7% and SQuAD v1.1 test f1 of 93.2. The limitations are insufficient fine-tuning and overfitting and limitations on input length there is a possible enhancement in domain specific pre-training and cross-lingual pre-training.

In [2], titled Towards Automatic Speech to Sign Language Generation, the main purpose of this model is to solve the highly challenging task of generating continuous sign language videos solely from speech segments for the first time. The techniques used in this paper are Speech to text models, Cross modal discriminator and transformer network, The model used transformer network to generate signer's poses. The metrics used are DTW(Dynamic Time Wrapping ) of 14.05 and PCK(Probability Correct Keypoints) of 53.33. The limitations are lack of efficient annotation criteria for text to sign language mappings and a proper dataset can be used for the model.

The authors of [3] proposed a model titled Translating Speech to Indian Sign Language Using Natural Language Processing. The model aims to develop Indian Sign language system that can be used for better communication with hearing impaired people. The techniques used are Tokenization, Lemmatization, Parsing and Part-of-speech tagging. Architecture in the study uses HMM. The evaluation metric is Net Promoter Score = 83.27. The drawbacks are Lack of robust sign language dataset, Necessary movements to be included for efficient use of NLP to sign language mapping.

In [4], the authors proposed a 3D avatar-based ISL learning model that can perform sign movements not only for isolated words but also for complete sentences through input text or speech. The proposed model in this paper takes input either English speech or text which is processed using a text processing technique to obtain ISL representation and next a gesture model is used to perform the sign movement corresponding to ISL with the help of an avatar. The proposed model in this paper has 3 phases, first it converts speech to English sentence for this they have used IBM-Watson service to convert input speech into text second translation of ISL sentence from English Sentence for this they used Natural Language Toolkit(NLTK), and finally generation of sign movements based on the input text is accomplished with the help of animation tool called Blender The evaluation parameters used are SER(sign error rate) of 10.50%, BLEU(Bilingual Evaluation Understudy) of 82.30%, NIST(National Institute of Standards and Technology) of 86.80%. The limitations are model is developed on limited corpus and no facial expressions.

In [5], authors developed a Gesture generation by robotic hand for aiding speech and hard of hearing persons based on Indian sign language. In this proposed model the interpreter system is designed with the help of robotic hand model and is programmed using Raspberry Pi4. The ISL dataset selected 28 different English alphabets each represented by various hand gestures. Raspberry Pi4 is used for speech recognition then the speech signal is processed and goes from Raspberry Pi4 to PCA9685 driver and next stage the translation from speech to sign and finally sign display by left and right robotic hand. The evaluation metric used is accuracy which evaluates the success of robotic hand interpreter by calculating mean and standard deviation, the best success rate is highest accuracy of 96%. The limitation is the generation of signs by the robotic hand is also influenced by background noise.

In [6], authors present an efficient isolated speech to sign conversion based on adaptive rate processing the proposed model in this paper dynamically adjusts processing parameters and uses KNN classification and sign conversion. The dataset used in paper involves isolated speech words consists of 5 male and 5 female each recorded 30 times and 600 utterances. In this model the speech is captured by microphone and processed and undergo adaptive rate filtering and then feature extraction by ARSTFT and classification by KNN. Proposed model achieved average subject dependent isolated accuracy of 96.6%. The limitations are they have included limited number of speakers for dataset and feature extraction limitation might not capture all relevant characteristics.

In [7], the authors aimed to create a real-time system that allows creating 3D sign language avatars with the help of neural networks. They used a dataset of thousands of sentences, which were associated with sign language gestures. The architecture consisted of a recurrent neural network (RNN) using long short-term memory (LSTM) schemes that captured temporal dependencies in spoken language. The highest performance result reported was 85% efficacy in producing the right sign language gestures corresponding to given input speech. Neural networks have the ability to adaptively learn from large datasets, and its real-time processing ability is also a significant advantage. This dataset was lacking in diversity, such that it might severely limit the effectiveness of the model across all dialects of sign language.

In [8], the authors were directed toward using motion capture technology in conjunction with deep learning techniques to enhance the accuracy of sign language recognition. Exhaustive dataset comprises of a multiple number of signs prepared by using different gestures of sign language, recorded at various environmental conditions. They used CNN architecture for processing motion data. They were able to demonstrate a jump of 90% in the recognition accuracy in controlled settings. It incorporates motion capture technology, which gives a rich data set of capture that outlines nuances within sign language, which ultimately leads to better recognition. Dependency on custom hardware and the motion capture would not be accessible. Moreover, the performance in uncontrolled environments still was not optimal.

In[9], the authors were testing the multimodal approaches in hopes that they would increase conversion from spoken language to sign language. They were using audio and visual along with textual data by amalgamating transformers and their attention mechanism in architecture. The dataset had a set of spoken language inputs with a corresponding sign gesture. The results of the evaluation provided a performance improvement overall, and the model produced an accuracy score of 87%. This multi-disciplinary approach can better contextualize the understanding and, hence, allow for more natural sign generation. This model makes the algorithm more complex, and hence its real-time application becomes difficult. Representational diversity of the dataset still needs to be increased.

In[10] , authors tackle the challenge of generating continuous sign language videos directly from speech, marking a significant advancement in assistive technology for the hearing-impaired. Previous methods have focused on translating text to sign language, but this approach does not fully capture the richness of speech, including emotions and contextual information. To address this gap, the authors propose a method that directly generates sign language from

natural, continuous speech, eliminating the need for text input. They collect and release the first large-scale Indian Sign Language dataset with speech-level annotations, text transcripts, and sign language videos. The core innovation is a multi-task transformer network that generates sign poses from speech segments in an end-to-end manner, using speech-to-text as an auxiliary task and a cross-modal discriminator for improved generation.

A Forward Step in Speech to Sign Language Translation Using the 3D Avatar Animator, the work here establishes a system for transforming English voice into Indian sign language animation (ES2ISL) for hearing-impaired people in India and others. The system, which makes the combination of Natural Language Processing (NLP), the Google Speech Recognizer API, and a predefined Indian Sign Language (ISL) database, speaks English and receives ISL animations. The model under discussion will significantly improve the communication experience for hearing-impaired people and have 77% accuracy with processing time of approximately 0.85 seconds [11].

Sign Language Transformers: Joint End-to-end Sign Language Recognition and Translation proposes a new methodology for joint Continuous Sign Language Recognition (CSLR) and Sign Language Translation (SLT), using transformer networks. Using a multi-task framework, the model learns both tasks in end to end manner, without the need for ground truth timing information. The proposed system combines recognition and translation into a single system, significantly boosting performance by utilizing a CTC loss function. Evaluation on the RWTH-PHOENIX-Weather-2014T dataset shows this method is outperforming previous approaches; delivering state-of-the-art results [12].

Speech to Indian Sign Language Translator project intends to develop a system that translates English audio into Indian Sign Language (ISL) after converting the audio to text, parsing the text to find its ISL grammatical equivalent. This system works on reordering the sentences and transforming them according to grammar and structure of ISL unlike all present systems where word to word mapping exists from input to output sign language. Using grammar rules, stemming, lemmatization and dictionary of sign language videos, the processed sentence can be converted into its appropriate ISL signs [13].

BERT (Bidirectional Encoder Representations from Transformers) proposes a new way of representing languages by pre-training deep bidirectional models which condition on both the left and the right context. Unlike previous models, like ELMo or OpenAI GPT, which are

unidirectional, BERT employs a masked language model (MLM) objective in order to fuse bidirectional context. Additionally, a next sentence prediction task is used to pre-train text-pair relationships [14].

Speech to text conversion and sentiment analysis on speaker specific data (2021) opened up internal core integration of speech recognition (SR) and sentiment analysis (SA) towards analyzing speakers' emotions. It discusses algorithms to determine advanced models of the speaker's emotional state. It extends past sentiment analysis applications that are restricted to reviews in films and social media analysis into speech based emotional sentiments detection. The proposed system combines speech-to-text conversion with sentiment analysis, utilizing techniques like tokenization, stemming, lemmatization, and polarity determination for emotion recognition [15].

## **2.3 Gaps Identified**

In previous systems for speech-to-sign language translation, several critical gaps have been identified, limiting their functionality, adaptability, and real-world applicability. These shortcomings include the following:

- **Lack of Multilingual Support:** Most existing systems are designed to work in one language, generally English, thereby greatly limiting the usage of such systems among various populations. Citing these limitations, the systems will be inaccessible to even those who do not know English or to those who use regional languages, thus excluding a bigger mass of potential users.
- **Noise and Multi-Speaker Support Missing From A Speech-to-Text\*\*:** The systems have great difficulty in transcribing speech in noise, as well as in situations of concurrent speech by several speakers. Whenever systems malfunction due to an absence of noise-rejection techniques or failing to reject other voices and instead mistaking them for underlying speech, they arrive out with wrong text, stopping mutual comprehension.
- **Lack of Real Time Video Output:** Previous systems have often failed to give a timely sign languages video report. These slow or still outputs greatly compromise their usability in dynamic real-world situations such as conversations, classrooms, that demand quick response.



- **Limited Support for Different Sign Languages:** Most systems work with just one sign language, usually American Sign Language (ASL), completely ignoring the different sign languages around the world (e.g., British Sign Language (BSL), Indian Sign Language (ISL), etc.). As a result, these systems will fail in multicultural environments or in regions where these variations apply.

The gaps pointed out above indicate the need for an inclusive, robust, and agile solution that considers all these drawbacks for its effective and wide acceptance across various socio-linguistic and cultural environments.

## **2.4 Problem Statement**

Communication between hearing and deaf individuals often requires interpreters or written text, which is a method highly prone to inefficiency and limits the degree of interaction occurring between these two groups. None of the current available solutions, like sign language interpreters, is always available at the time of need, and moreover, text-based communication often lacks expressiveness and usually requires a lot of time.

This vacuum of accessible and efficient translation tools leads to serious limitations in the possibility for hearing-impaired people to fully take part in social, educational, and professional environments. This innovative project seeks to effectively address the significant need for an advanced automated system that translates spoken language into sign language in real time. This groundbreaking technology will facilitate and enable more inclusive communication between individuals, ensuring that everyone, regardless of their hearing abilities, can engage and interact seamlessly with one another.

## **2.5 Proposed Solution**

A proposed solution, the system enables fast speech-to-sign translation. This will benefit people who suffer from hearing defects in that they will be able to talk and communicate with those who have the ability to hear. The system shall comprise of the following major components: conversion of speech to text, processing the text by NLP, and sign language movement creation that shall be rendered as video to the user.

The first step is to change spoken words into text using special speech recognition models like Whisper. This module can work with many languages, making it good for people who don't

speak English. The speech-to-text system works well with different accents, speech speeds, and background noise, so it can be used in many different places.

Once the speech is written down as text, NLP techniques like those with the BERT and T5 models work on the extracted text. The system identifies and keeps track of all important keywords and ideas to ensure that it captures the main message correctly. The translation part would then change the text to follow the grammar and structure rules of sign language.

**Sign Language Movements and Animation** The processed keywords that are matched belong to existing sign language gestures coming from two popular sign language datasets. In case some of the movements do not exist in the existing datasets, then uniqueness or special terms can be created with new gestures by using tools like Blender. This brings about animated gestures that are conveyed by a virtual avatar and communicated clearly and easily. The modular design of the system allows flexibility and growth. New languages can be added, new sign language data can be integrated, and the ever-evolving needs of communication can be met in this system. This particular solution provides both robust NLP and animation techniques that help hearing-impaired individuals communicate better through it.

## **2.6 Summary**

Solution involves developing a marginally real-time speech to sign language conversion system. The system involves conversion of speech to text, processing of extracted text i.e identification of key words using NLP, and display of corresponding sign language movements as a video to end user. Speech to text will be incorporating multiple languages and not just English. Sign language movements of two majorly used sign languages will be used from existing datasets. New movements will be constructed (if required) using Blender

## **3. System Requirements Specification**

### **3.1 Introduction**

#### **3.1.1 Purpose of the requirements document**

This Software Requirements Specification (SRS) document defines and identifies the functional and non-functional requirements of the system for the Speech to Sign Language Translation using 3D Model. The system takes the advantage of advanced Speech Recognition and Natural Language Processing(NLP) methods so as to provide real-time sign language for users who are deaf, thus bridging communication gap between challenged and normal users. This document is supposed to explain to the reader on the core features of the product and how each part of it will work to achieve its purpose appropriately and effectively.

The SRS document will act as a guideline for all developers, designers engaged in the project. Clear descriptions of the use case and design elements in the document will be such that the meaning of the system is ensured; it would help in identifying other tools and technologies needed, speech recognition algorithms, real-time video rendering, and translation algorithms.

#### **3.1.2 Scope of the product**

##### **3.1.2.1. Real-Time Speech-to-Sign Language Translation**

The core functionality of the product is to capture spoken input, convert it to text, and map the text to sign language gestures. This involves multiple stages:

The project consists of the following functionalities:

- Transcription of speech using NLP-based speech-to-text systems.
- Conversion of the processed text to a sign language gesture through a mix of key word extraction and mapping rules adhering to sign language grammar and syntax.
- Animation of a realistic 3D avatar to perform the sign language gestures in a smooth, expressive, and accurate way.

As such, the objective behind this bundle of functionality is to build a strong system translating spoken language into visually perceived sign language in real time so as to reduce the dependency on human interpreters and fill the gaps of communication.

##### **3.1.2.2. Contextual and Idiomatic Translation**

For the purpose of going beyond simple translation, this system intends to apply more sophisticated NLP techniques to discern context, meaning, and idiomatic expressions. In other words, it aims to:

- **Context-Aware NLP Models:** Ensure that translations carry meaning and appropriateness based on an understanding of the context of conversations.
- **Idiomatic Expression Handling:** Making use of idiomatic expressions and translations of idiomatic expression into words and expressions that sign the equivalent for natural and fluent communication.

The above aspects of the product mean that more accurate contextual translation will occur, which is important for effective communication in real life.

### **3.1.2.3. User Feedback and Continuous Learning**

The system will have a mechanism for feedback to allow users to tag the accuracy and quality of translation. Such feedback could be directly used in the improvement of translation models and the ability to adapt to users' needs continuously. Core items include:

- **User Correction and Feedback Loop:** Correct users' suggestions to improve the errors in translations so that the same can improve in the long-run.
- **Adaptive Learning Models:** The feedback will be used along with machine learning to rectify any inaccuracies and improve the translation in the realization of phrases or special terms.

This feedback would ensure that the system becomes more and more reliable and accurate in use.

### **3.1.2.4. Augmented Reality (AR) Integration**

Future versions of the system may see an Augmented Reality (AR) capability overlaying signing avatars in actual environments where they are interactive and useful. AR-integrated platforms would allow real-world communication assistance for the user.

## **3.1.3 Definitions, Acronyms, and Abbreviations**

To clarify technical language used in this document, key terms are defined below:

- **NLP:** Natural Language Processing, used for text extraction and language processing.
- **3D Model:** A computer-generated representation of a character used for sign language gestures.
- **HCI:** Human-Computer Interaction, a field focusing on the design and use of computer technology.
- **AR:** Augmented Reality, a technology that overlays virtual elements in the real world.
- **ISL:** Indian Sign Language, an example of a regional sign language.

- BLEU: Bilingual Evaluation Understudy, a metric for evaluating translation quality.
- PCK: Probability Correct Keypoints, a metric for evaluating pose estimation accuracy

### **3.1.4 References**

[1] IEEE, 2009. 830-1998-IEEE Recommended Practice for Software Requirements Specifications. IEEEExplore.

### **3.1.5 Overview of the remainder of the document**

This SRS is organized into a logical flow from a high level concept of the product, down to very detailed technical requirements. In the Section 2, we describe the product's perspective, key functions, target user groups, operational constraints and dependencies. This section provides a basic understanding of how the system combines an advanced speech recognition and in real time feedback to help Users of all skill levels communicate through sign language in a safe and efficient way.

Section 3 presents our specific functional and non-functional requirements which include core functionalities of user authentication, speech recognition, live video rendering, natural language processing, and real time language translation. Usability, reliability, and accessibility are emphasized as non-functional requirements, usability and performance also, so a user can have a seamless and responsive experience. In addition, there is a description of the main screens and the interactions that users will encounter.

The document strikes a balanced mode between technical precision and understandable explanations, enabling all stake holders to become aware of the necessities and effectively participate in developing this interactive yoga training platform.

## **3.2 General Description**

### **3.2.1 Product Perspective**

The speech-to-sign language converter is the real-time integration of voice recognition, natural language processing, and either 3D animation or gesture representation to help enable effective communication from spoken language users to sign language users. The system receives spoken words or phrases and converts those through voice recognition into text; this is then subjected to NLP analysis to fully contextualize and get the intent meaning, such that correct sign language is translated.

After processing the spoken language with a 3D avatar or on-screen gesture animations, the translated signs appear on the computer screen. The 3D avatar resembles a human signer who acts out the correct gestures of the phrases that are identified, and the on-screen animations offer visual expressions of the signs. The main objectives of this system are to enhance accessibility for deaf and hard-of-hearing people to the world of communication, with inclusivity and free flow in various settings, be it educational, workplaces, or social.

### **3.2.2 Product Functions**

#### **3.2.2.1 Voice Recognition**

Voice recognition is the initial component of the speech-to-sign language converter, capturing spoken words or phrases in real-time and converting them into text. This technology employs advanced algorithms to analyze audio input, enabling it to recognize various accents, dialects, and speech patterns for high accuracy. By processing continuous speech, the system allows for seamless communication in both quiet and noisy environments. This foundational capability ensures that the spoken language is accurately represented in text format for further processing.

#### **3.2.2.2 Sign Language Translation**

Once the spoken words are converted to text, the system processes this text to identify corresponding signs or gestures in the target sign language. This translation goes beyond simple word-for-word conversion; it incorporates an understanding of context, grammatical structures, and nuances of both languages. Using natural language processing techniques, the system selects the most appropriate signs, ensuring that the intended meaning is conveyed accurately. This approach allows the system to handle idiomatic expressions and emotional subtleties effectively.

#### **3.2.2.3 Visual Representation**

The translated signs are visually represented through two primary methods: a 3D avatar or gesture animations displayed on-screen. The 3D avatar mimics human movements, providing a relatable and engaging representation of sign language gestures. Alternatively, on-screen gesture animations may be simpler visual cues that quickly convey the signs. This visual output is crucial for facilitating communication, allowing users to see and understand the signs in real-time.

#### **3.2.2.4 User Feedback**

To enhance user interaction, the system includes a clear and user-friendly interface for feedback. Users can easily pause, replay, or adjust the gestures as needed, allowing for a better learning experience. This feature is especially beneficial for those who are new to sign language, enabling them to practice and refine their skills. A responsive interface ensures accessibility for individuals with varying levels of technical proficiency.

### **3.2.2.5 Customization Options**

The system also offers customization options to cater to diverse user needs. Individuals can select different sign language dialects, ensuring that regional variations are accurately represented. Users can adjust the speed of the sign language gestures to suit their comprehension levels, enhancing their viewing experience. Additionally, accessing translation history allows users to review previously translated phrases, reinforcing their understanding of both spoken and signed languages. These features contribute to a more personalized and effective communication tool, promoting inclusivity and accessibility.

### **3.2.3 User Characteristics**

**Primary Users:** The primary users of the speech-to-sign language converter encompass a diverse group, including deaf or hard-of-hearing individuals, teachers, interpreters, and hearing individuals interested in learning sign language. Deaf and hard-of-hearing individuals rely on this technology to facilitate communication in various settings, such as schools, workplaces, and social environments, bridging the gap between spoken and signed languages. Teachers can use the system to create more inclusive classroom environments, ensuring that all students, regardless of their hearing ability, can participate fully in discussions and learning activities.

Interpreters may utilize the system to enhance their interpretation skills or assist in real-time translations during events. Finally, hearing individuals who wish to learn sign language can benefit from the system as a valuable educational tool, helping them grasp the fundamentals of sign language through visual representation.

**Moderate grasp on technology:** The user skills required to engage with the speech-to-sign language converter can range from basic to moderate understanding of technology. While users may have varying degrees of familiarity with sign language, the system is designed to be intuitive and user-friendly for both individuals with and without prior knowledge of signing. Users with a background in sign language will find the system valuable for reinforcing their skills, while those new to signing can easily navigate the interface and engage with the

translated content.

### 3.2.4 General Constraints

#### **High accuracy in real-time pose estimation:**

These algorithms, MoveNet and YOLOv8 then determine the outcome of the performance. Good in performance, however, these algorithms may fail sometimes to detect some poses due to other factors like noisy keypoints, and changing lighting conditions, camera angles, or yogic poses that may end up looking similar. Eventually, wrong pose classification may cause bad feedback that could negatively impact the users' experience or, in extreme circumstances, harm them.

#### **Hardware Requirements:**

The user should have an equipment that supports camera facilities like a smartphone, tablets, laptops, and other types of webcams that can capture real-time video. It should support huge processing hardware that can be run continuously in real time, while analyzing the video feed. As far as the lower-end devices are concerned, not much may be possible: it may either experience a lag while giving feedback or poor image resolution and a system crash during long use.

#### **Internet connectivity:**

The speed at which the internet is used would depend on the users who rely on the cloud for processing. Many poses and feedback depend significantly upon the transfer rate between the user's device and server. Users relying on slow and unreliable connections to the internet would bring forth several issues like latency feedback delay that would invalidate the concept of real-time correction. The system, hence, is limited in its application to certain settings since it is very dependent on the internet's speed and its reliability.

### 3.2.5 Assumptions and Dependencies

- **Speech Recognition Accuracy:** Assumes that speech recognition algorithms can accurately capture spoken input, even in noisy environments.
- **Sign Language Database:** Relies on an extensive, reliable database of sign language gestures to cover vocabulary and context.
- **User Device Specifications:** Assumes that users have compatible hardware and sufficient processing capabilities.



- **Cloud Services:** Assumes that third-party APIs (for speech-to-text and possibly NLP) are accessible and dependable for reliable operation.
- **User Familiarity:** Assumes that users are familiar with the basics of sign language if they wish to verify translation accuracy.

### **3.3. Specific Requirements**

#### **3.3.1 Functional Requirements**

##### **FR\_3.1.1. User Authentication**

- FR\_3.1.1.1. Login: The system shall allow users to log in using a username and password for access to the application.
- FR\_3.1.1.2. Sign Up: The system shall enable new users to create an account by providing a name, email, and password.

##### **FR\_3.1.2. Speech Input and Recognition**

- FR\_3.1.2.1. Speech Capture: The system shall capture real-time spoken language input from the user's microphone.
- FR\_3.1.2.2. Voice-to-Text Conversion: The system shall convert the captured speech into text format using speech recognition technology.

##### **FR\_3.1.3. Sign Language Translation**

- FR\_3.1.3.1. Text Processing: The system shall process the converted text to identify corresponding sign language gestures.
- FR\_3.1.3.2. Contextual Understanding: The system shall use natural language processing (NLP) to analyze and interpret the context and meaning of the speech for accurate translation.

##### **FR\_3.1.4. Visual Sign Language Representation**

- FR\_3.1.4.1. 3D Avatar Animation: The system shall display sign language translations through a 3D avatar that performs the appropriate gestures.
- FR\_3.1.4.2. Gesture Animation: The system shall offer gesture animations as an alternative option to visually represent sign language translations.

##### **FR\_3.1.6. Customization Options**

- FR\_3.1.6.1. Sign Language Dialect Selection: The system shall allow users to choose different sign language dialects (e.g., ASL, BSL) based on their preference.
- FR\_3.1.6.2. Speed Adjustment: The system shall enable users to adjust the speed of sign language gesture playback.

### 3.3.2 Non-Functional Requirements:

**NFR\_3.2.1. Usability:** The system shall be a user-friendly interface, which includes easy navigation across the platform. Irrespective of the technical knowledge of the user any person shall be able to utilize the system in a smooth manner. The interfaces shall be clear, responsive and support recording and file uploads.

**NFR\_3.2.2. Performance:** The system shall provide real-time response and low latency to the users for real-time translation from speech to sign language. Audio capture and processing for sign language shall be smooth to enhance the user's experience.

**NFR\_3.2.3. Environmental Constraints:** The system shall work optimally in any environment: whether changing noise levels or multiple users for giving the correct sign language. Optimized for both noisy and silent backgrounds, it can handle background noise that would otherwise interfere with the voice prompts.

**NFR\_3.2.4. Reliability:** The system shall exhibit high availability and have minimal downtime across a whole year to ensure constant access to the users. In order to guarantee integrity of data, accuracy in translation, key-word extraction and sign language mapping.

### 3.3.3 User Interface Requirements

#### Login/Sign-Up Page

- Textboxes for email and password are provided.
- Buttons for login and sign up will be visible on the webpage.

#### Home Page

- It contains option to record video or attach an audio file.
- A button to end the recording or conversation.
- Rendered video is displayed in the same page.
- Option to download the video is also provided once video is made available.
- New conversation can be started by hitting on record again.

### 3.3.4 Hardware Requirements

Hardware Requirements for Speech-to-Sign Language Conversion System:

#### 3.3.4.1. GPU (Graphics Processing Unit) - NVIDIA RTX 3080 or Higher

The need for a GPU is paramount for such computationally demanding tasks like speech

recognition, natural language processing (NLP), and real-time animations of sign language gestures. Models such as Whisper, BERT, and T5 relate to very large-scale matrix operations and deep learning computations, which are greatly accelerated with current-day GPUs. Therefore, it includes a high-performance model such as NVIDIA RTX 3080 in order to process faster and achieve near-real time performance for the task of converting speech to sign language.

#### **3.3.4.2. CPU: Central Processing Unit: Intel Core i7 (12th Gen) or AMD Ryzen 7 5800X or better**

Reason: The overall orchestration of the system, be it data preprocessing, integration of different modules, and running background logics, is handled by the CPU. To carry out such tasks as audio preprocessing, real-time system responses, and coordinating computations using the GPU (with algorithms such as noise reduction with RNNs), one would need a powerful multi-core CPU. A high clock speed and multiple cores ensured the seamless execution of processes.

#### **3.3.4.3. RAM: 32 GB DDR4 or Higher**

Reason: Models with deep learning such as Whisper, BERT, T5 consume large memory for loading the model weights, for intermediate computation, and also pre-processing large size datasets and real time video outputs. Processing multiple audio streams or large-scale sign language animations might require at least 32 GB of RAM to prevent memory bottlenecks.

#### **3.3.4.4. Storage: 1 TB SSD**

Reason: SSDs guarantee high read and write data rates - necessary for fast loading of huge datasets and models. This project deals with datasets for speech, text, and sign language gestures, along with pre-trained models for speech-to-text and NLP. A 1 TB SSD ensures enough space and rapid files access, which reduces system latency.

#### **3.3.4.5. Dedicated Sound Card or External Audio Interface**

Reason: Quality sound cards or external audio interfaces are essential for clean audio input processing, especially in noisy environments, to be able to cancel background noises with reasonable clarity to do the transcription right.

#### **3.3.4.6. Monitor: 1080p or Higher Resolution with 60Hz Refresh Rate**

Reason: A high-resolution monitor is required for visualization and testing of animated sign

language gestures. A refresh rate of 60Hz will ensure smooth output in video, as required for the debugging and testing of the animation in real-time.

#### **3.3.4.7. Webcam and Microphone**

Reason: A good quality microphone will capture the input speech for transcription. Minimize the noise in the background. Web camera can be used for lip reading incorporation, if necessary, or real-time user feedback during testing.

These are the minimum equipment requirements to ensure the system delivers optimal performance to sustain the real-time computations of a multilingual, multi-module speech-to-sign language conversion application.

### **3.4 Architecture and UML Diagrams**

#### **3.4.1 Architecture Diagram**

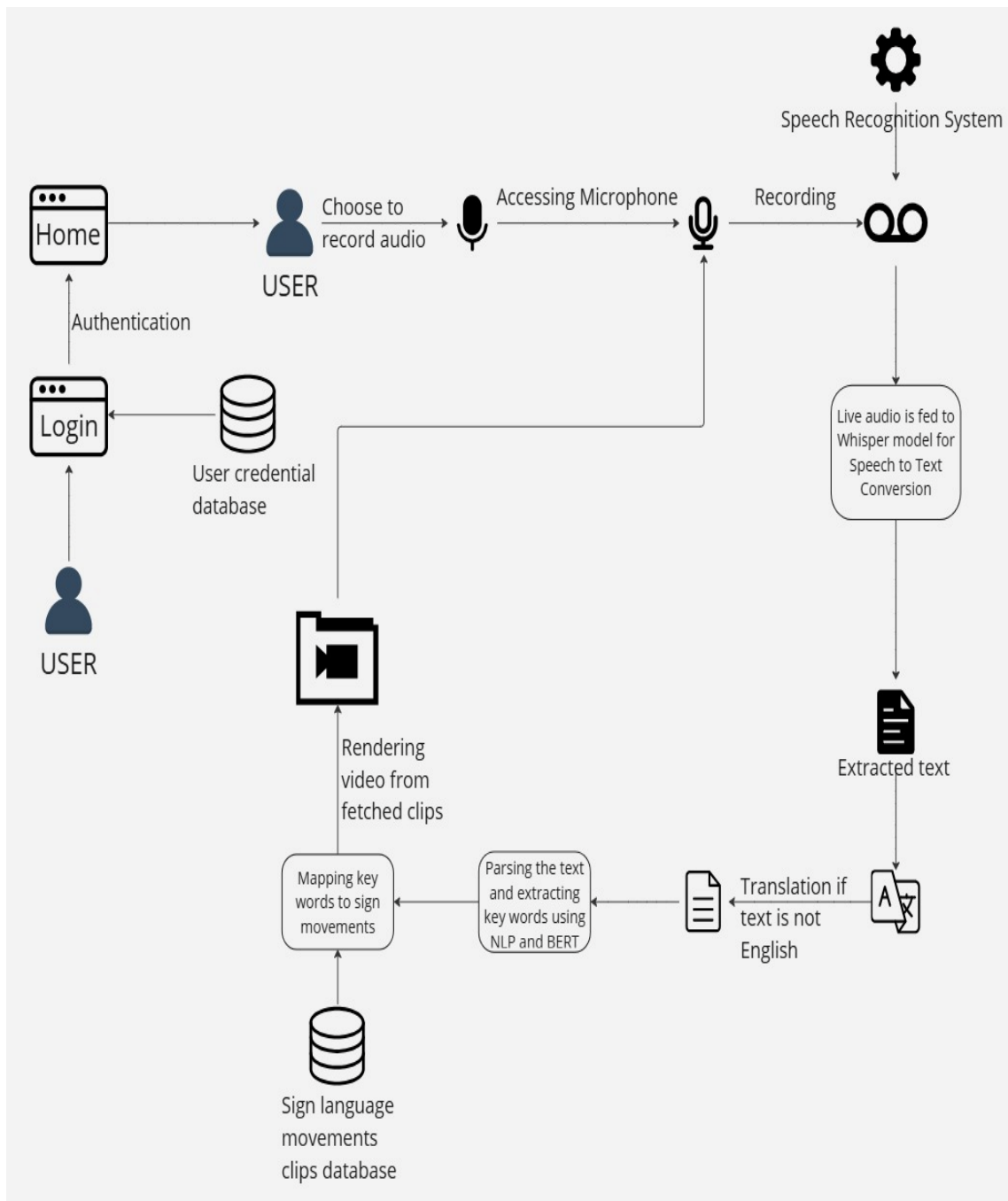


Fig 1. Architecture Diagram

### 3.4.2 Use case Diagram



Fig 2. Use Case Diagram

### 3.4.3 Class Diagram

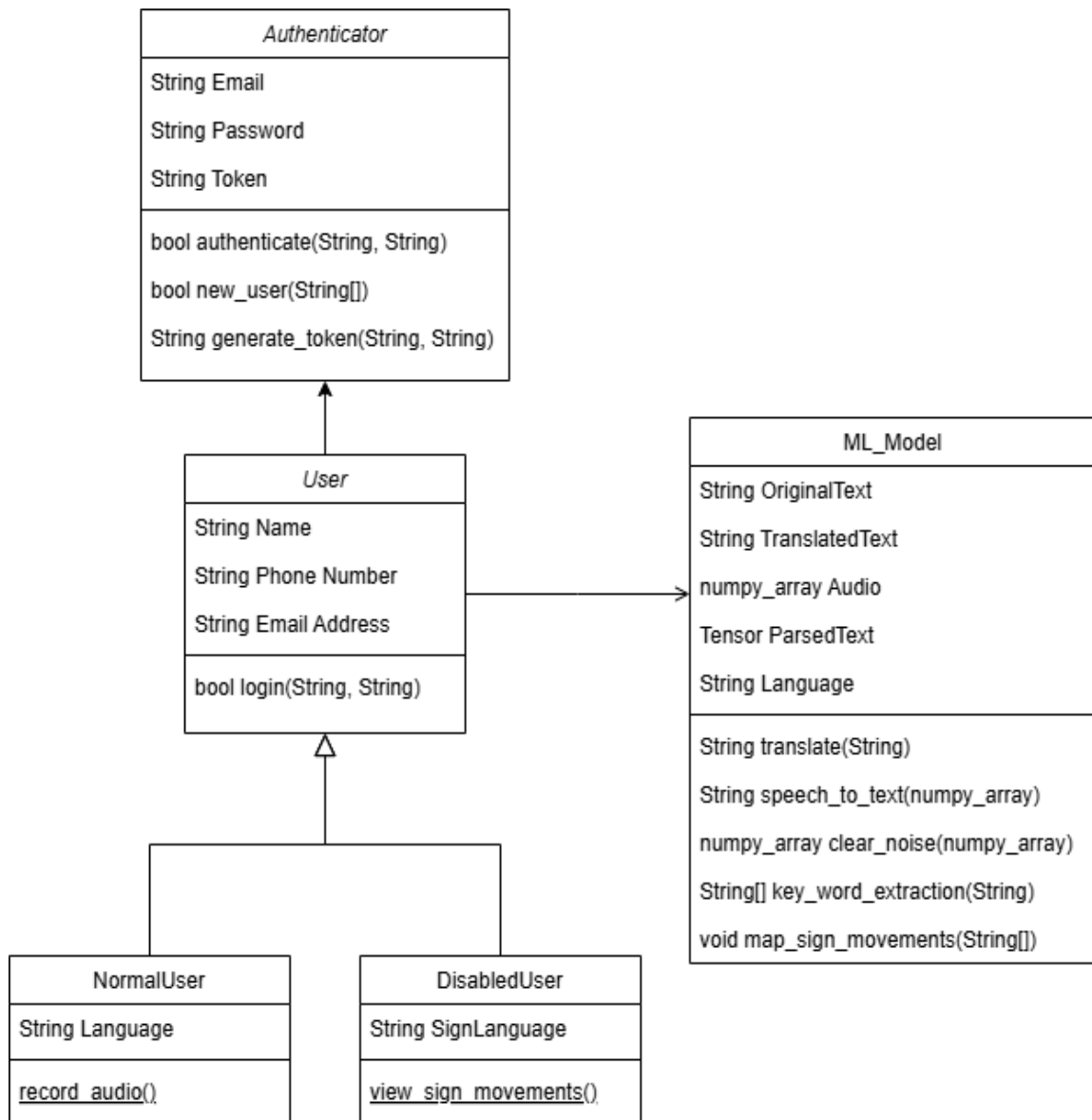


Fig 3. Class Diagram

### 3.4.4 Activity Diagram

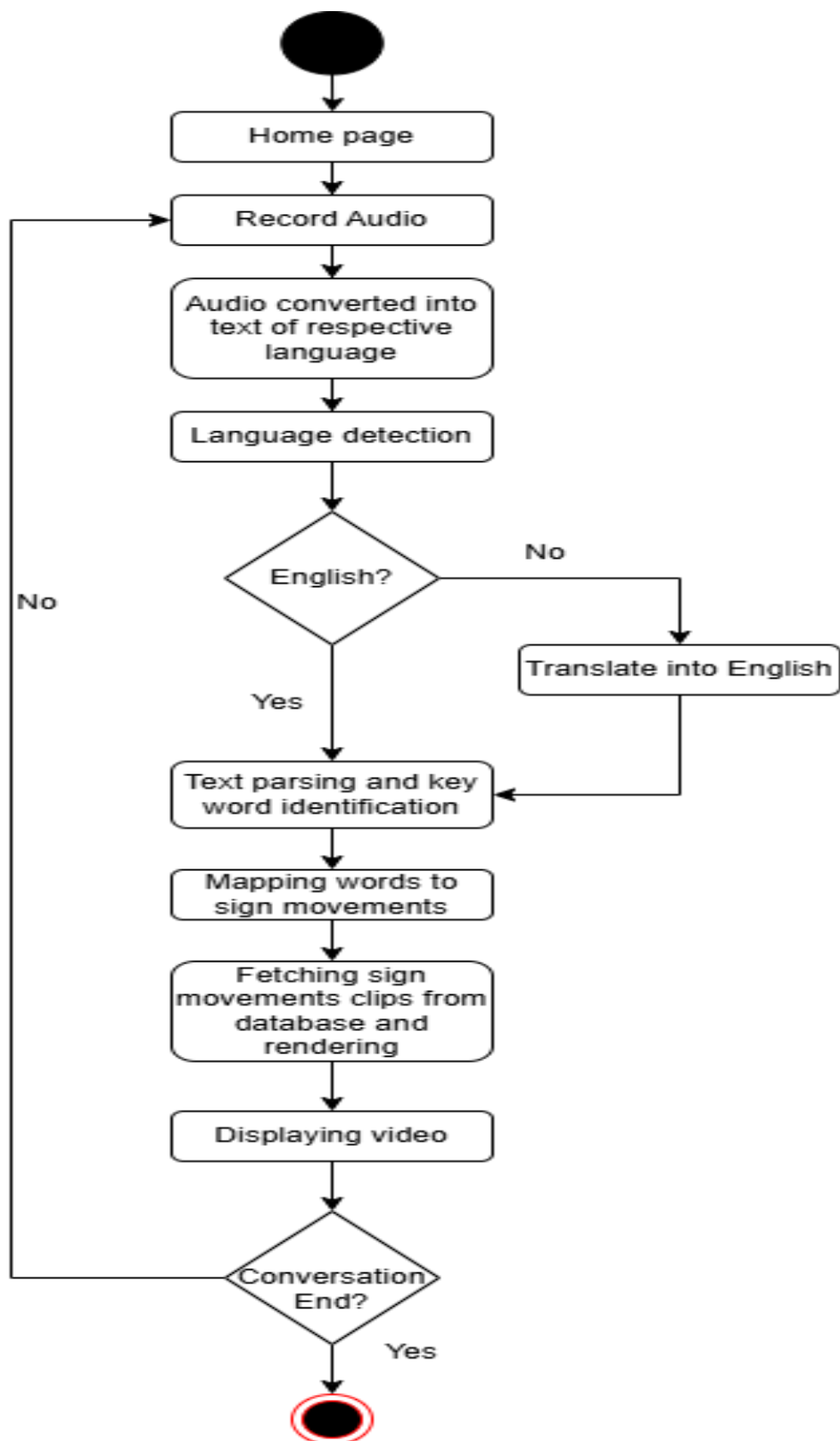


Fig 4. Activity Diagram

### 3.4.5 Flow Chart



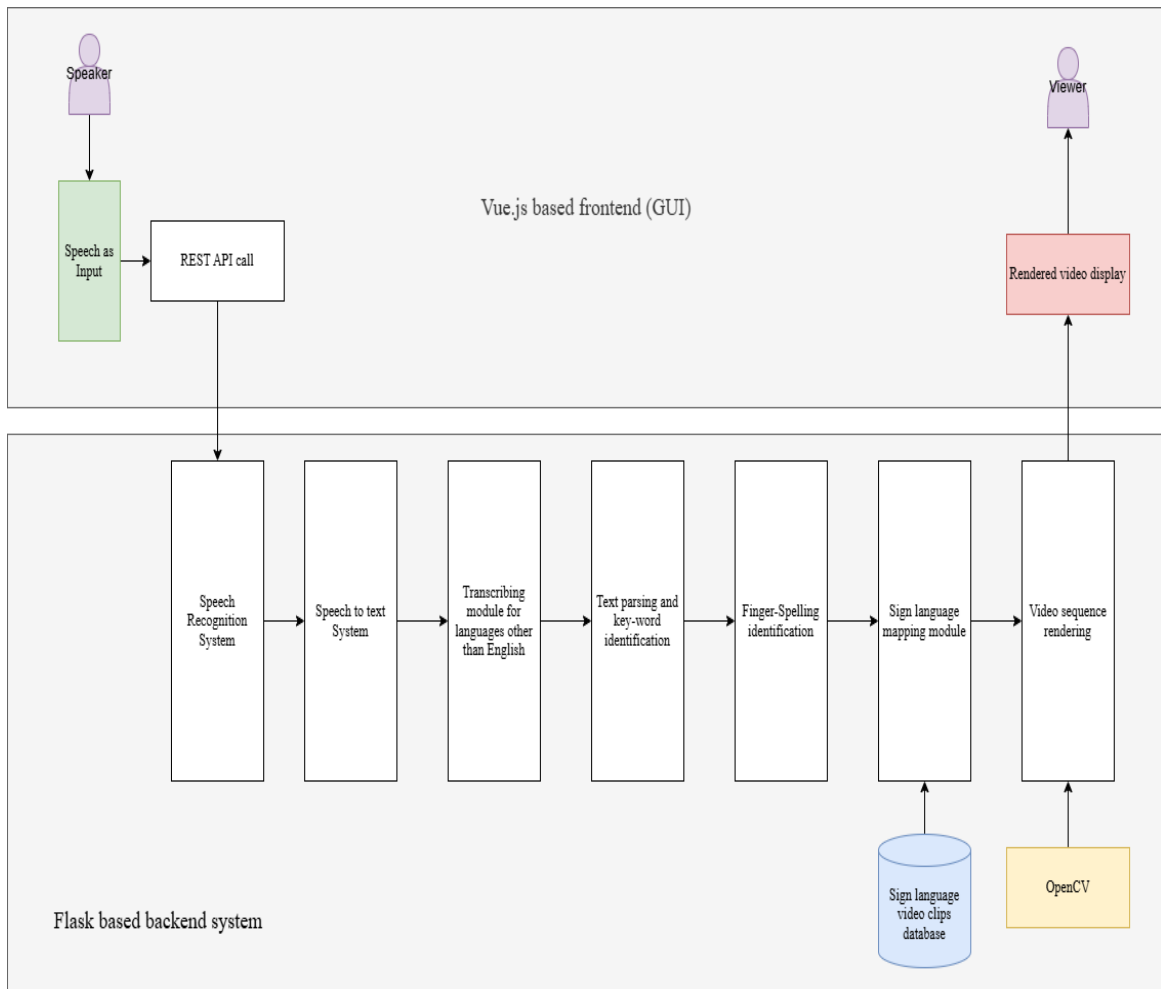


Fig 5. Flow Chart

### 3.5 Summary

The project is designed to bridge communication between disabled and normal people in real time conversations. The section details on the requirements, modules, UML designs, constraints and assumptions which will be relied on for future product development.

## 4. Methodology

## 4.1 Module Description

A suggested architecture of the system is to have several mutually dependent modules providing a high level of accuracy and reliability in the system. The modules consist of some advanced AI-based models, which are oriented towards specific sub-tasks, as described in the sections below.

### 4.1.1. Automated speech-to-text transcription (Whisper Model)

This module uses OpenAI's Whisper model, which is one of the best speech recognition models. Here are some benefits of using Whisper:

Resilience to variability: Successfully accommodates a range of accents, fluctuations in speech rate, and multilingual use.

- Noise tolerance: Shows useful performance in actual environments that have ambient noise.
- Multilingual support: Enhances versatility by supporting transcription across multiple languages.

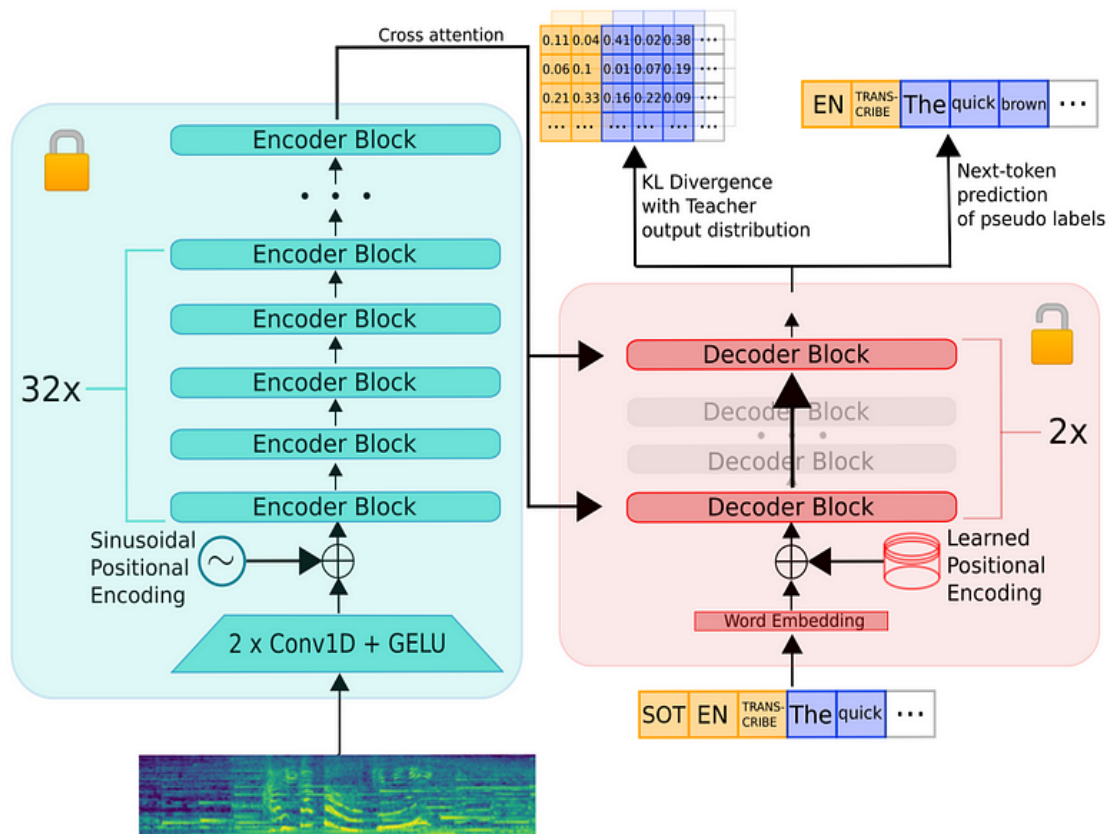


Fig 6. Architecture of Whisper Model

### 4.1.2. Background Noise Cancellation using Recurrent Neural Networks – RNN

This module applies RNNs for suppressing the presence of undesirable background noise from audio streams.

Process the data sequentially; therefore, RNNs are excellently suited for handling time-series data and thereby are very suitable for capturing temporal dependencies in audio signals.

The denoised audio provides better clarity and enhances the transcription accuracy in the Whisper model.

Dynamic noise adaptation: The system is able to adapt for varying levels of noises experienced in different environments.

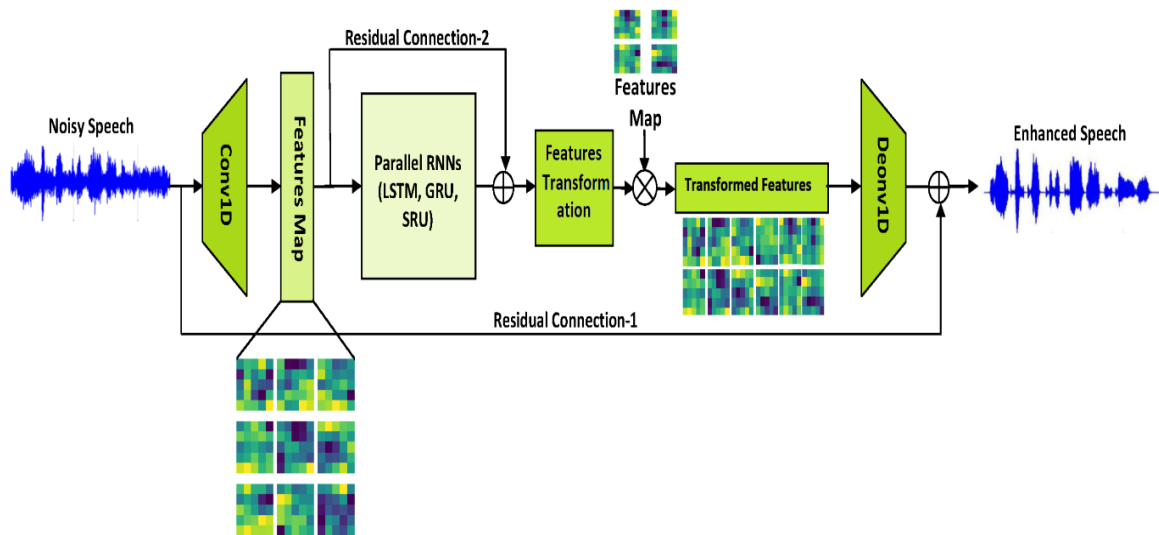


Fig 7. Architecture of RNN used for denoising

#### 4.1.3. Translate Text Google T5 Model

- The Google T5 is a Text-to-Text Transfer Transformer model that converts transcribed text to more understandable sign language.
- Morpho-syntactic transformation: A transformation that presents the text according to the sign linguistic and grammatical norms.
- Pretrained functionality: Provides the minimal training requirement using pre-trained natural language processing modules while providing accurate translations.
- Output can be fine-tuned to support different domain-specific terminologies and context.

#### 4.1.4. Keyword Extraction- BERT: Bidirectional Encoder Representations from Transformers

The keywords that are semantically and contextually relevant from the translated text are identified using BERT.

Understanding context: It relates to understanding the connotation and nuances of the text to ensure accurate choice of words.

- Emphasize fundamental elements: Aids in prioritizing key terms and expressions that require emphasis during sign language gestures.
- Greater clarity: It ensures that the intended message is effectively communicated through the translation.

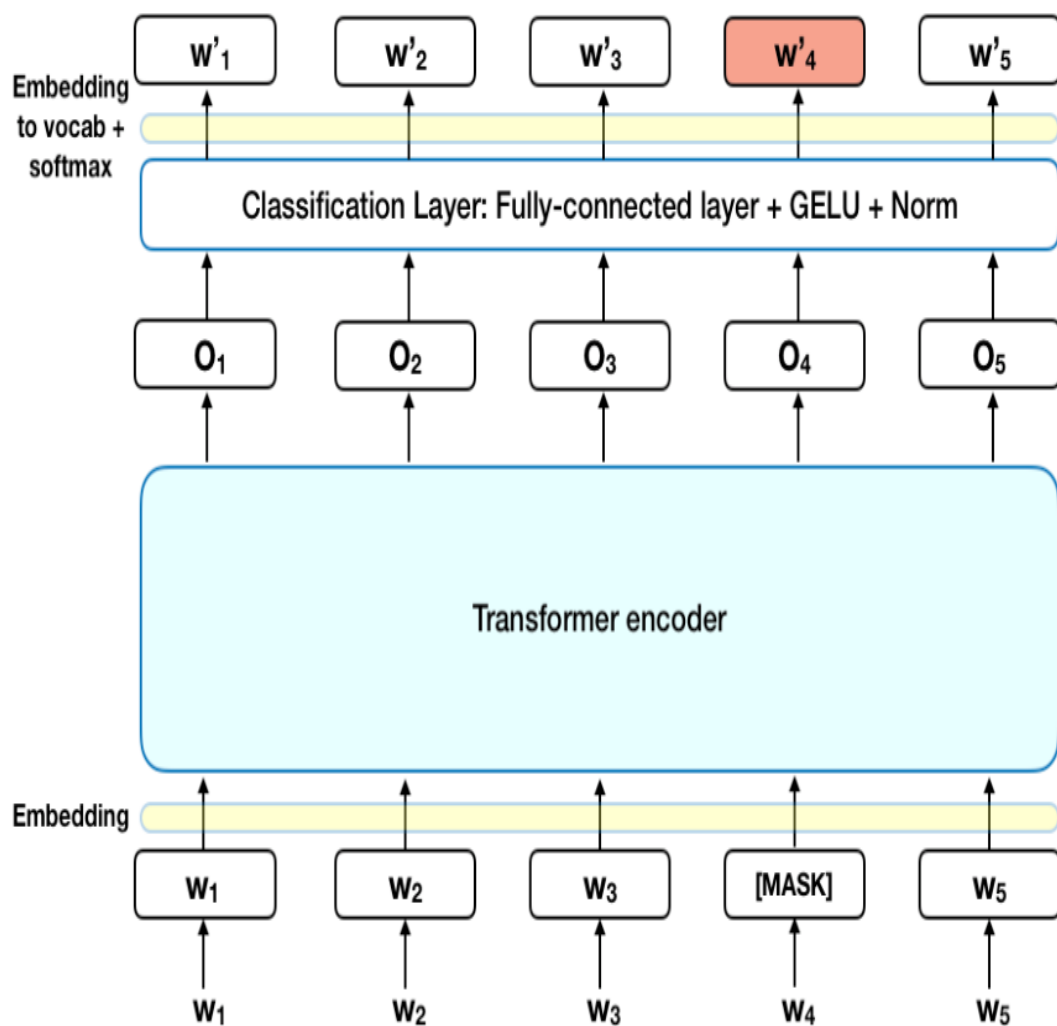


Fig 8. Architecture of BERT

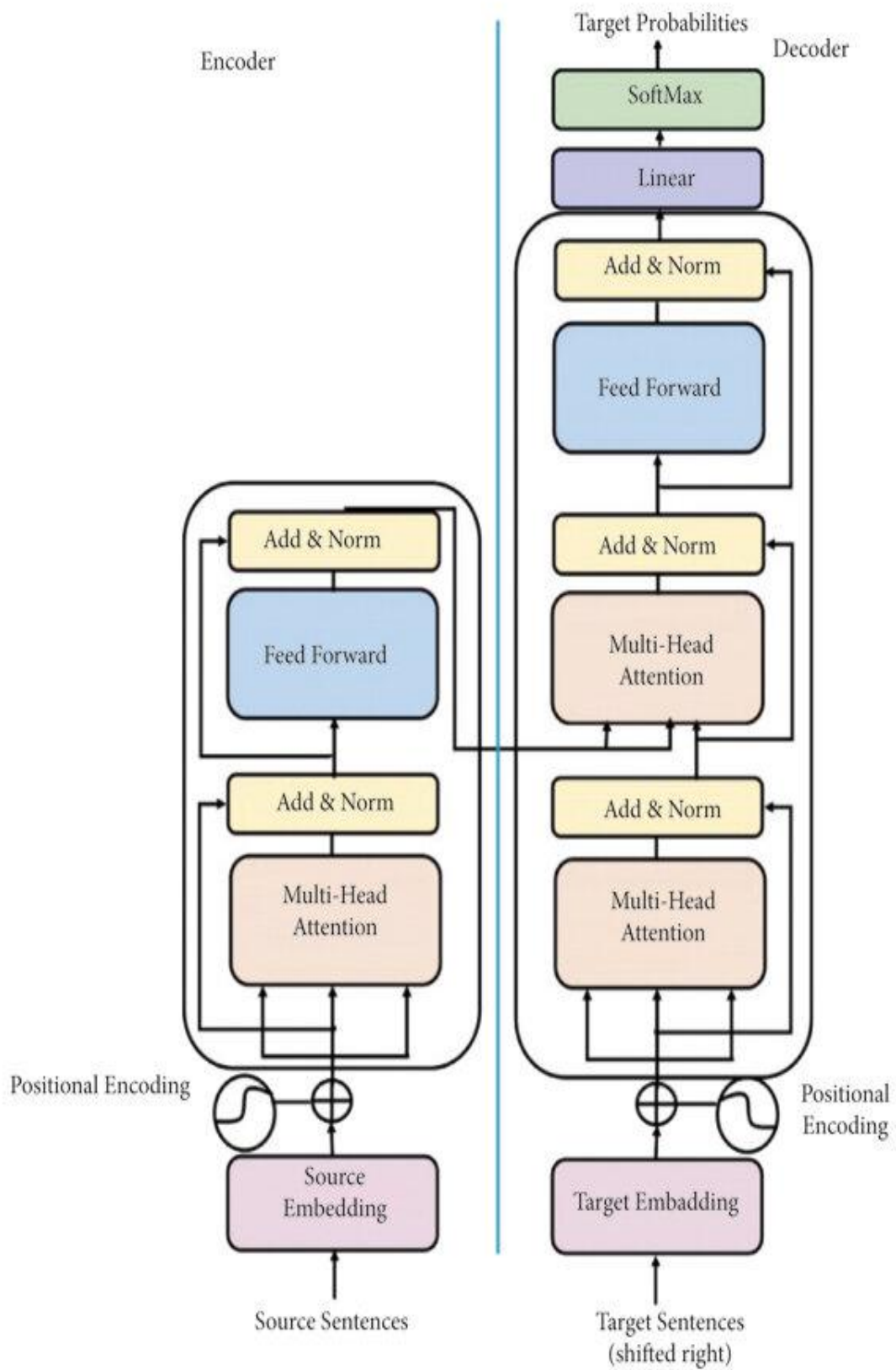


Fig 9. Transformer architecture

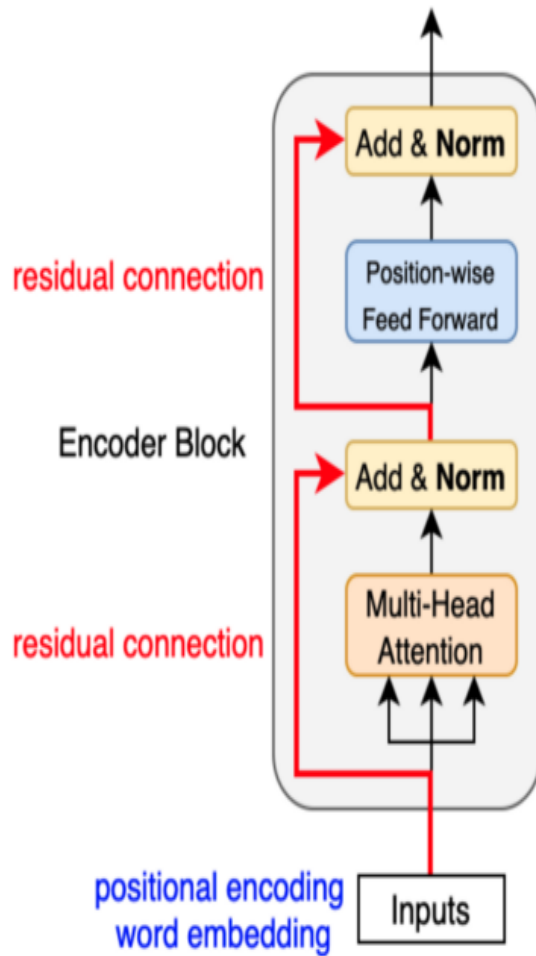


Fig 10. Architecture of Encoder used in BERT

#### 4.1.5. Hand Gesture Mapping and Animations

It translates the sensed words to gestural sign language, using a lexicon of gestures prestored.

- Lexicallexicon of gestures: Symbols for key words are correlated together, thus more straightforwardly translating.
- This is made interactive and visually clear with virtual avatar animation, in which gestures will be animated.
- Customization: It can offer region-specific signs or language-specific signs for other users.

## 4.2 Implementation Methodology Designed for Solution End

### 4.2.1. Data Collection

Collect large amounts of audio recordings using various accents, dialects, speech rates, and noise conditions (done in pre-trained whisper model). Prepare text datasets especially for translating sign language, such as phrases most used in conversations. Gather a library of sign

language gestures annotated with corresponding keywords or phrases.

#### **4.2.2. Preprocessing**

Audio preprocess: Improve the de-noising using RNN-based noise cancellation module to suppress background interference to improve audio clarity.

Text preprocessing generally covers normalizing, tokenization, and any textual cleansing to ensure the effectiveness of input in further modules. Dataset preparation: Label and organize data for efficient training and testing workflows.

#### **4.2.3. Speech-to-Text Translation**

Used Whisper to transcribe audio whose text is transcribed. Measured the quality of the transcription in terms of WER and fine-tune models for specific environments to determine robustness over variations in languages and dialects.

#### **4.2.4. Text Translation**

Feed the transcription into the Google T5 model, aiming to translate it into a simplified and sign-language-compatible format. Implement the fine-tuning mechanisms to deal with nuances in translation, which would particularly be valuable for idiomatic expressions. Above all, ensure that the output text adheres to sign language grammar.

#### **4.2.5. Keyword Extraction**

Apply BERT to find keywords and concepts of importance from the filtered text. Pay special attention in extracting pragmatic and contextually relevant phrases for avoiding ambiguity in sign gestures. Tune parameters for BERT based on the subtleties possible in texts concerning sign language.

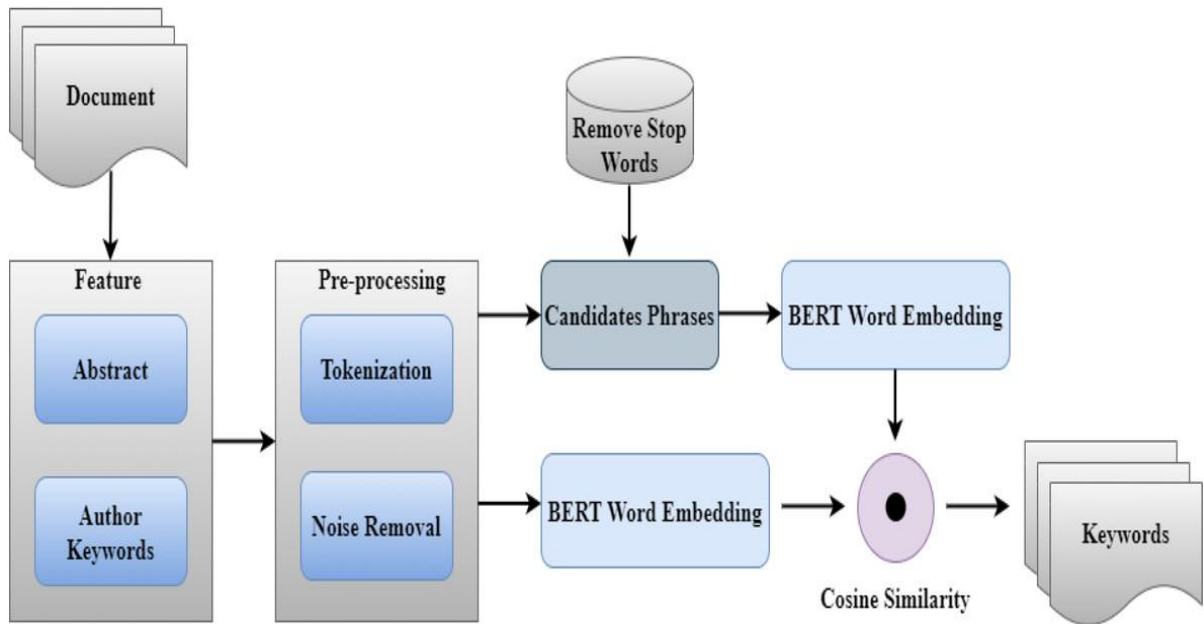


Fig 11. BERT for keyword extraction.

#### 4.2.6. Mapping Keyword to sign movements

Mapping Sign Language Pair the identified key words with appropriate gestures found in the standard vocabulary of gestures. Apply animation technology and frameworks of animation to visually describe those gestures in an avatarmode interface. Accommodate regional variations in signed languages by providing multiple libraries of gestures.

### 4.3 Implementation

#### 4.3.1 Pre-built libraries and frameworks

Open-source and free of charge, Kivy promotes rapid application development under the Python programming language for applications with multitouch user interfaces. With a single codebase, developers can deploy applications for a seamless experience on Windows, macOS, Linux, Android, and iOS. Kivy supports widgets, layout, graphics rendering - OpenGL ES 2, touch inputs, and more, providing great help to multimedia applications such as music-media players, drawing applications, and audiovisual interaction systems.

For an application with audio, anything can be done in Kivy: designing the user interface; programming buttons to start/playback, volume sliders, and so on; and, in addition, display visualizations of sound waves or spectrograms.

By using threads in Python, you can execute multiple operations at the same time; this is particularly important in applications with long-running tasks, such as audio streaming,



recording, or doing something else in real-time without blocking the UI. Because the Kivy UI runs on a main thread, any long-running operation, which could be such as reading audio input or processing that input, should be transferred to a different thread to keep the interface responsive.

For example, an audio recorder built with Kivy would continuously read audio data from a microphone with a second thread while allowing the GUI to refresh or take user input.

### **4.3.2 Code**

#### **4.3.2.1 app.py**

```
import os
import sqlite3
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button
from kivy.uix.label import Label
from kivy.uix.textinput import TextInput
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.uix.videoplayer import VideoPlayer
from kivy.clock import Clock
import sounddevice as sd
import numpy as np
import wave
import threading
import platform
import subprocess
from datetime import datetime
from functools import partial
from speech_to_text import main_text

class Database:
    def __init__(self):
        self.db_path = "user_database.db"
        self.initialize_db()
```

```

def initialize_db(self):
    conn = sqlite3.connect(self.db_path)
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            mobile TEXT UNIQUE,
            password TEXT,
            created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    """)
    conn.commit()
    conn.close()

def user_exists(self, mobile):
    conn = sqlite3.connect(self.db_path)
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM users WHERE mobile = ?", (mobile,))
    user = cursor.fetchone()
    conn.close()
    return user is not None

def validate_login(self, mobile, password):
    conn = sqlite3.connect(self.db_path)
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM users WHERE mobile = ? AND password = ?",
        (mobile, password))
    user = cursor.fetchone()
    conn.close()
    return user is not None

def register_user(self, mobile, password):
    try:
        conn = sqlite3.connect(self.db_path)

```

```

        cursor = conn.cursor()

        cursor.execute("INSERT INTO users (mobile, password) VALUES (?, ?)",
                        (mobile, password))

        conn.commit()

        conn.close()

        return True

    except sqlite3.IntegrityError:

        return False

    except Exception as e:

        print(f"Registration error: {str(e)}")

        return False

# Login Screen

class LoginScreen(Screen):

    def __init__(self, db, **kwargs):

        super().__init__(**kwargs)

        self.db = db

        layout = BoxLayout(orientation='vertical', padding=20, spacing=10)

        title = Label(text='Audio Video App Login', font_size=24, size_hint_y=0.2)

        form = GridLayout(cols=2, spacing=10, size_hint_y=0.4)

        form.add_widget(Label(text='Mobile Number:'))

        self.mobile_input = TextInput(multiline=False, input_type='number', input_filter='int')

        form.add_widget(self.mobile_input)

        form.add_widget(Label(text='Password:'))

        self.password_input = TextInput(multiline=False, password=True)

        form.add_widget(self.password_input)

        buttons_layout = BoxLayout(orientation='horizontal', size_hint_y=0.2, spacing=10)

        self.login_btn = Button(text='Login')

        self.login_btn.bind(on_press=self.verify_login)

        self.register_btn = Button(text='Register')

        self.register_btn.bind(on_press=self.go_to_register)

```

```

buttons_layout.add_widget(self.login_btn)
buttons_layout.add_widget(self.register_btn)
self.error_label = Label(text="", color=(1, 0, 0, 1), size_hint_y=0.2)
layout.add_widget(title)
layout.add_widget(form)
layout.add_widget(buttons_layout)
layout.add_widget(self.error_label)
self.add_widget(layout)

def verify_login(self, instance):
    mobile = self.mobile_input.text.strip()
    password = self.password_input.text.strip()

    if not mobile or not password:
        self.error_label.text = "Please enter both mobile number and password"
        return

    if not mobile.isdigit():
        self.error_label.text = "Mobile number should contain only digits"
        return

    if self.db.validate_login(mobile, password):
        self.manager.transition.direction = 'left'
        self.manager.current = 'home'
        self.mobile_input.text = ""
        self.password_input.text = ""
        self.error_label.text = ""
    elif self.db.user_exists(mobile):
        self.error_label.text = "Invalid password"
    else:
        self.error_label.text = "User not found. Please register."

def go_to_register(self, instance):
    register_screen = self.manager.get_screen('register')

```

```

register_screen.mobile_input.text = self.mobile_input.text

self.manager.transition.direction = 'left'

self.manager.current = 'register'

self.error_label.text = ""

# Registration Screen

class RegisterScreen(Screen):
    def __init__(self, db, **kwargs):
        super().__init__(**kwargs)

        self.db = db

        layout = BoxLayout(orientation='vertical', padding=20, spacing=10)

        title = Label(text='Register New Account', font_size=24, size_hint_y=0.2)

        form = GridLayout(cols=2, spacing=10, size_hint_y=0.5)

        form.add_widget(Label(text='Mobile Number:'))

        self.mobile_input = TextInput(multiline=False, input_type='number', input_filter='int')

        form.add_widget(self.mobile_input)

        form.add_widget(Label(text='Password:'))

        self.password_input = TextInput(multiline=False, password=True)

        form.add_widget(self.password_input)

        form.add_widget(Label(text='Confirm Password:'))

        self.confirm_password_input = TextInput(multiline=False, password=True)

        form.add_widget(self.confirm_password_input)

        buttons_layout = BoxLayout(orientation='horizontal', size_hint_y=0.2, spacing=10)

        self.register_btn = Button(text='Register')

        self.register_btn.bind(on_press=self.process_registration)

        self.back_btn = Button(text='Back to Login')

        self.back_btn.bind(on_press=self.go_back_to_login)

        buttons_layout.add_widget(self.register_btn)

        buttons_layout.add_widget(self.back_btn)

        self.error_label = Label(text="", color=(1, 0, 0, 1), size_hint_y=0.2)

        layout.add_widget(title)

        layout.add_widget(form)

        layout.add_widget(buttons_layout)

```

```

layout.add_widget(self.error_label)
self.add_widget(layout)
def process_registration(self, instance):
    mobile = self.mobile_input.text.strip()
    password = self.password_input.text.strip()
    confirm_password = self.confirm_password_input.text.strip()

    if not mobile or not password or not confirm_password:
        self.error_label.text = "Please fill in all fields"
        return

    if not mobile.isdigit():
        self.error_label.text = "Mobile number should contain only digits"
        return

    if len(mobile) < 10:
        self.error_label.text = "Mobile number should be at least 10 digits"
        return

    if password != confirm_password:
        self.error_label.text = "Passwords don't match"
        return

    if len(password) < 6:
        self.error_label.text = "Password should be at least 6 characters"
        return

    if self.db.user_exists(mobile):
        self.error_label.text = "Mobile number already registered"
        return

    if self.db.register_user(mobile, password):
        self.error_label.text = "
        self.manager.transition.direction = 'left'
        self.manager.current = 'home'
        self.mobile_input.text = "
        self.password_input.text = "

```

```

        self.confirm_password_input.text = "
else:
    self.error_label.text = "Registration failed. Please try again."
def go_back_to_login(self, instance):
    self.manager.transition.direction = 'right'
    self.manager.current = 'login'
    self.error_label.text = "
class AudioVideoInterface(BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.orientation = 'vertical'
        self.padding = 20
        self.spacing = 10
        self.speech_to_video_function = main_text
        self.top_section = BoxLayout(
            orientation='vertical',
            size_hint_y=0.3,
            spacing=10
        )
        self.status_label = Label(
            text='Ready to record',
            size_hint_y=0.3
        )
        self.record_btn = Button(
            text='Start Recording',
            size_hint_y=0.3
        )
        self.record_btn.bind(on_press=self.toggle_recording)
        self.top_section.add_widget(self.status_label)
        self.top_section.add_widget(self.record_btn)
        self.video_player = VideoPlayer(
            size_hint_y=0.7,

```

```

        state='stop',
        options={'allow_stretch': True}
    )
    self.add_widget(self.top_section)
    self.add_widget(self.video_player)
    self.is_recording = False
    self.frames = []
    self.sample_rate = 44100
    self.video_path = None
    self.recordings_dir = "recordings"
    if not os.path.exists(self.recordings_dir):
        os.makedirs(self.recordings_dir)
def toggle_recording(self, instance):
    if not self.is_recording:
        self.start_recording()
    else:
        self.stop_recording()
def start_recording(self):
    self.is_recording = True
    self.frames = []
    self.record_btn.text = 'Stop Recording'
    self.status_label.text = 'Recording...'
    threading.Thread(target=self.record_audio).start()

def stop_recording(self):
    self.is_recording = False
    self.record_btn.text = 'Start Recording'
    self.status_label.text = 'Processing...'
    threading.Thread(target=self.save_and_process_audio).start()

def record_audio(self):
    def callback(indata, frames, time, status):

```



```

        if self.is_recording:
            self.frames.append(indata.copy())
    with sd.InputStream(channels=1, callback=callback,
                        samplerate=self.sample_rate):
        while self.is_recording:
            sd.sleep(100)
def save_and_process_audio(self):
    if not self.frames:
        self.update_status('No audio recorded')
        return
    try:
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        audio_filename = f"recording_{timestamp}.wav"
        audio_filepath = os.path.join(self.recordings_dir, audio_filename)
        audio_data = np.concatenate(self.frames, axis=0)
        with wave.open(audio_filepath, 'wb') as wf:
            wf.setnchannels(1)
            wf.setsampwidth(2)
            wf.setframerate(self.sample_rate)
            wf.writeframes((audio_data * 32767).astype(np.int16))
        # Process audio with external function
        self.update_status('Converting speech to video...')
        self.video_path = self.speech_to_video_function(audio_filepath)
        print(self.video_path)
        Clock.schedule_once(lambda dt: self.play_video(self.video_path))
    except Exception as e:
        self.update_status(f'Error: {str(e)}')
def play_video(self, video_path):
    if os.path.exists(video_path):
        self.update_status('Playing video externally...')

    if platform.system() == "Windows":

```

```

        os.startfile(video_path)

    elif platform.system() == "Darwin": # macOS
        subprocess.run(["open", video_path])
    else: # Linux
        subprocess.run(["xdg-open", video_path])

    Clock.schedule_once(lambda dt: self.delete_video(video_path), 10)

    else:

        self.update_status('Video file not found')

def delete_video(self, video_path):

    try:

        if os.path.exists(video_path):

            os.remove(video_path)

            print("File deleted successfully")

    except Exception as e:

        print(f"Error deleting file: {str(e)}")

def update_status(self, text):

    Clock.schedule_once(lambda dt: self.set_status(text))

def set_status(self, text):

    self.status_label.text = text

```

# Home Screen with Audio/Video functionality

```

class HomeScreen(Screen):

    def __init__(self, **kwargs):

        super().__init__(**kwargs)

        layout = BoxLayout(orientation='vertical', padding=20, spacing=10)

        header = BoxLayout(orientation='horizontal', size_hint_y=0.1)

        title = Label(text='Audio Video App', font_size=20)

        self.logout_btn = Button(text='Logout', size_hint_x=0.3)

        self.logout_btn.bind(on_press=self.logout)

        header.add_widget(title)

        header.add_widget(self.logout_btn)

        self.audio_video_interface = AudioVideoInterface()

```

```

        layout.add_widget(header)

        layout.add_widget(self.audio_video_interface)

        self.add_widget(layout)

    def logout(self, instance):

        self.manager.transition.direction = 'right'

        self.manager.current = 'login'

class AudioVideoApp(App):

    def build(self):

        self.db = Database()

        self.sm = ScreenManager()

        self.login_screen = LoginScreen(self.db, name='login')

        self.register_screen = RegisterScreen(self.db, name='register')

        self.home_screen = HomeScreen(name='home')

        self.sm.add_widget(self.login_screen)

        self.sm.add_widget(self.register_screen)

        self.sm.add_widget(self.home_screen)

        self.sm.current = 'login'

    return self.sm

    def on_stop(self):

    try:

        if os.path.exists("recordings"):

            for file in os.listdir("recordings"):

                if file.endswith((".mp4", ".wav")): # Clean up audio and video files

                    try:

                        file_path = os.path.join("recordings", file)

                        if os.path.exists(file_path):

                            os.remove(file_path)

                            print(f"Deleted: {file}")

                    except Exception as e:

                        print(f"Error deleting {file}: {str(e)}")

            except Exception as e:

                print(f"Error during cleanup: {str(e)}")

```

```
if __name__ == '__main__':
    AudioVideoApp().run()
```

#### 4.3.2.2 text\_to\_sign.py

```
import os

from moviepy import VideoFileClip, ImageClip, concatenate_videoclips
import cv2
import numpy as np

class SignLanguageCompiler:

    def __init__(self, signs_directory):
        self.signs_dir = signs_directory
        self.supported_extensions = {'.mp4', '.jpg', '.png', '.gif'}

    def find_file(self, name):
        name = name.capitalize()
        for ext in self.supported_extensions:
            path = os.path.join(self.signs_dir, f'{name}{ext}')
            if os.path.exists(path):
                return path
        return None

    def create_clip(self, file_path, duration=1.0):
        if file_path.endswith('.mp4'):
            clip = VideoFileClip(file_path)
        else:
            clip = ImageClip(file_path).set_duration(duration)
        return clip

    def spell_word(self, word):
        letter_clips = []
        for letter in word:
            if letter.isalpha():
```

```

        letter_path = self.find_file(letter)

        if letter_path is None:

            raise ValueError(f"Letter not found: {letter}")

        clip = self.create_clip(letter_path, duration=0.5)

        letter_clips.append(clip)

    return concatenate_videoclips(letter_clips) if letter_clips else None


def compile_sentence(self, sentence, output_path):

    words = ".join(c for c in sentence if c.isalnum() or c.isspace()).split()

    clips = []

    for word in words:

        sign_path = self.find_file(word)

        if sign_path:

            # Use existing sign video/image

            clip = self.create_clip(sign_path)

        else:

            # Spell out the word

            clip = self.spell_word(word)

        if clip: # Only add if we successfully created a clip

            clips.append(clip)

    if not clips:

        raise ValueError("No valid clips were created")

    final_video = concatenate_videoclips(clips)

    # Write the final video

    final_video.write_videofile(

        output_path,

        fps=30,

        codec='libx264',

        audio=False

    )

    final_video.close()

```

```

        for clip in clips:
            clip.close()
def main_sign(text):
    compiler = SignLanguageCompiler(signs_directory="signs/")
    for i in range(10):
        print(text)
    sentence = "I have an Apple"
    compiler.compile_sentence(text, "output_sign_language.mp4")
    return "output_sign_language.mp4"
if __name__ == "__main__":
    main_sign()

```

#### 4.3.2.3 speech\_to\_text.py

```

import torch
import whisper
from text_to_sign import main_sign
class WhisperSTT:
    def __init__(self, model_size='small'):
        self.device = 'cuda' if torch.cuda.is_available() else 'cpu'
        self.model = whisper.load_model(model_size).to(self.device)
    def transcribe(self, audio_path, language=None):
        options = {
            'fp16': torch.cuda.is_available(),
            'language': language
        }
        result = self.model.transcribe(audio_path, **options)
        return {
            'text': result['text'],
            'language': result['language'],
            'segments': result['segments']
        }
    def translate(self, audio_path):

```

```

options = {
    'task': 'translate',
    'fp16': torch.cuda.is_available()
}
result = self.model.transcribe(audio_path, **options)
return {
    'text': result['text'],
    'source_language': result['language']
}

def main_text(path):
    stt = WhisperSTT(model_size='small')
    result = stt.transcribe(path, language='en')
    print(result['text'])
    return main_sign(result)

```

## 4.4 Summary

Integration of Synthesis and Assessment: Harmonious integration of each module to ensure seamless flow, characterized by low latency between phases, is important in ensuring uninterrupted flow of information. Testing: Utilizing iterative testing protocols with various audio samples facilitates assessment of system performance in transcription accuracy, translation speed, and gesture accuracy. Real-world simulations: Real-world simulations conducted under different levels of background noise and different accents need to be tested. The envisioned architecture, being modular and scalable speech-to-sign language translation system, employs the latest models such as Whisper, RNNs, T5, and BERT. This architecture entails large-scale audio pre-processing, text translation, keyword extraction, and strict gesture mapping. This approach, therefore, fosters flexibility, sustainability, and adaptability, making it easy for hearing-impaired people to integrate into various linguistic environments. The modular architecture allows easy upgrading and scalability, thus positioning the system as an invaluable resource in fostering inclusive communication.

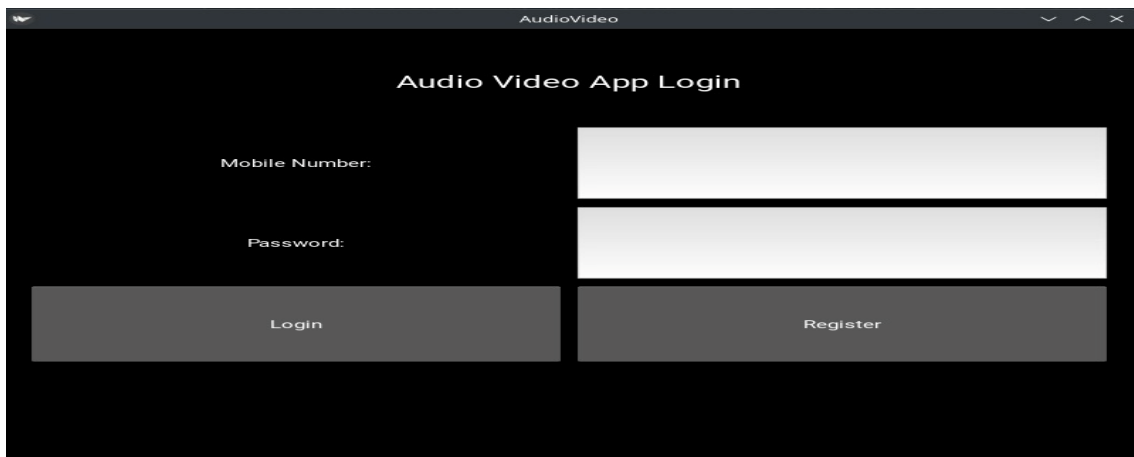
## 5. Testing and Results

### 5.1 Unit Testing

Test No.	Feature being tested	Expected Input	Expected Output	Actual Input	Actual Output	Result of the Test (Pass/Fail)
1	Login	Correct set of mobile and password	Successful login	Correct Set of credentials	Successful Login	Pass
2	Login	Correct set of mobile and password	Successful login	Wrong credentials	Error message	Pass
3	Registration	Mobile and Password	Successful Registration	New user credentials	Successful signup	Pass
4	Registration	Mobile and Password	Successful Registration	Existing user mobile	Error message	Pass
5	Logout	Click on logout button	Redirected to login page	Click on logout button	Successfully redirected to login page	Pass
6	Audio Recording	Record Audio by clicking on the record button	Temporary audio file is created and saved for further processing	Recorded audio	File is created and processing successful	Pass
7	Output video	Audio of user	Video of hand movement gestures	Recorded audio	Video created and played successfully	Pass

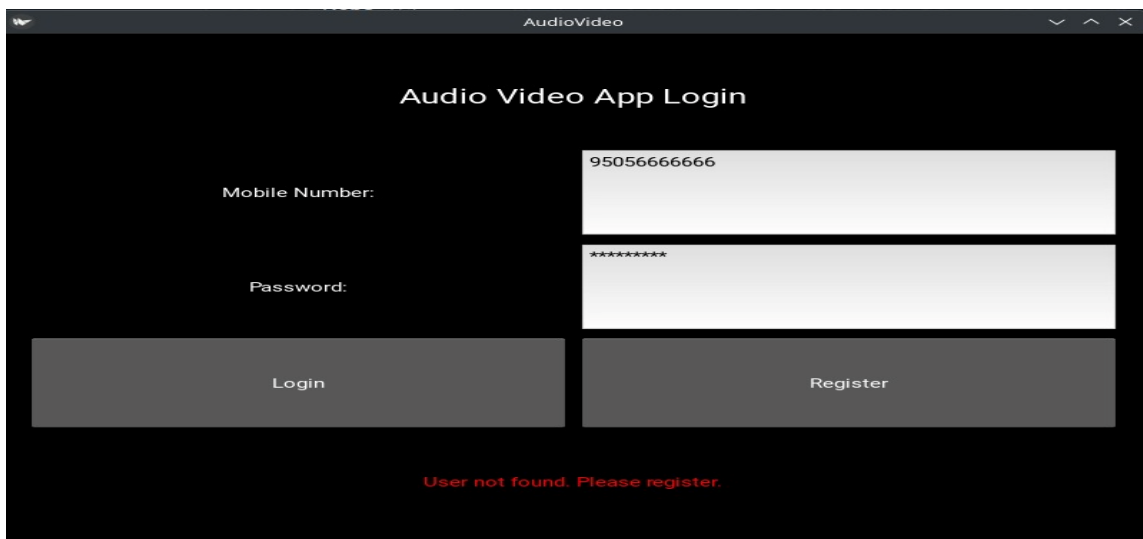


## 5.2 Results



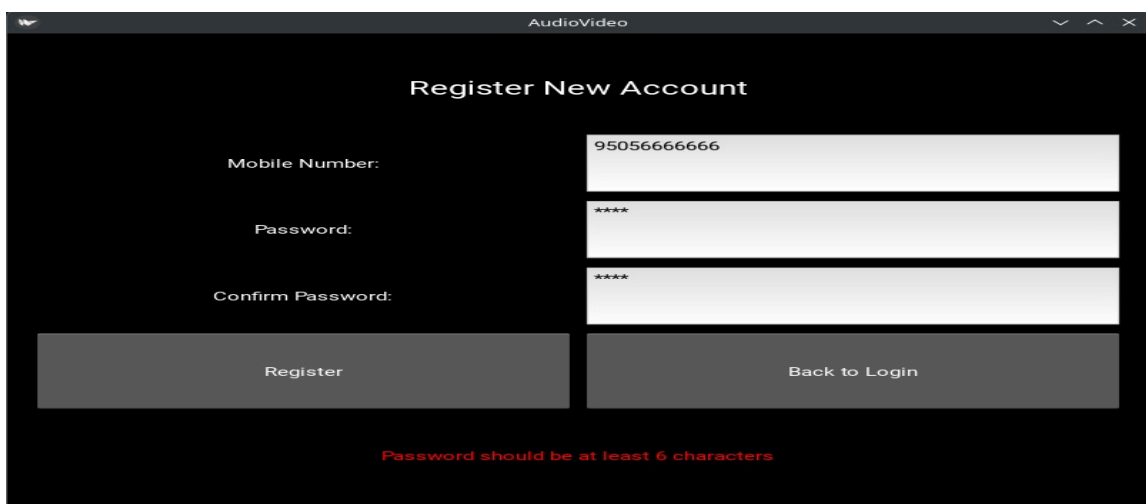
The screenshot shows a web application window titled "AudioVideo" with a dark background. The main heading is "Audio Video App Login". Below the heading, there are two input fields: "Mobile Number:" and "Password:". Below these fields are two buttons: "Login" and "Register".

Fig 12. User Login Page



The screenshot shows the same "Audio Video App Login" form as in Fig 12. The "Mobile Number:" field now contains the text "95056666666". The "Password:" field contains "\*\*\*\*\*". Below the buttons, a red error message is displayed: "User not found. Please register.".

Fig 13. Invalid User Login



The screenshot shows a web application window titled "AudioVideo" with a dark background. The main heading is "Register New Account". Below the heading, there are three input fields: "Mobile Number:", "Password:", and "Confirm Password:". Below these fields are two buttons: "Register" and "Back to Login". A red error message is displayed at the bottom: "Password should be at least 6 characters".

Fig 14. User Registration Form

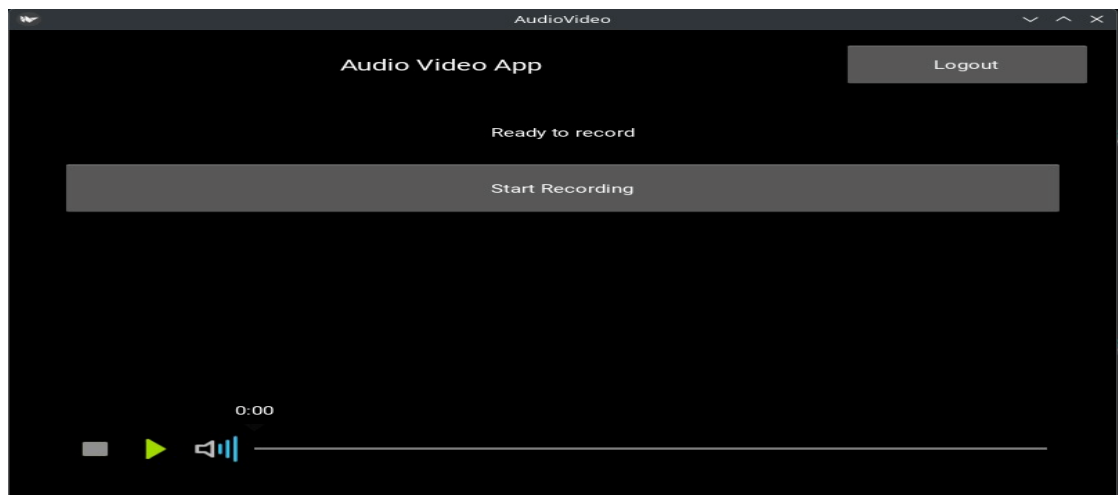


Fig 15. Application Dashboard

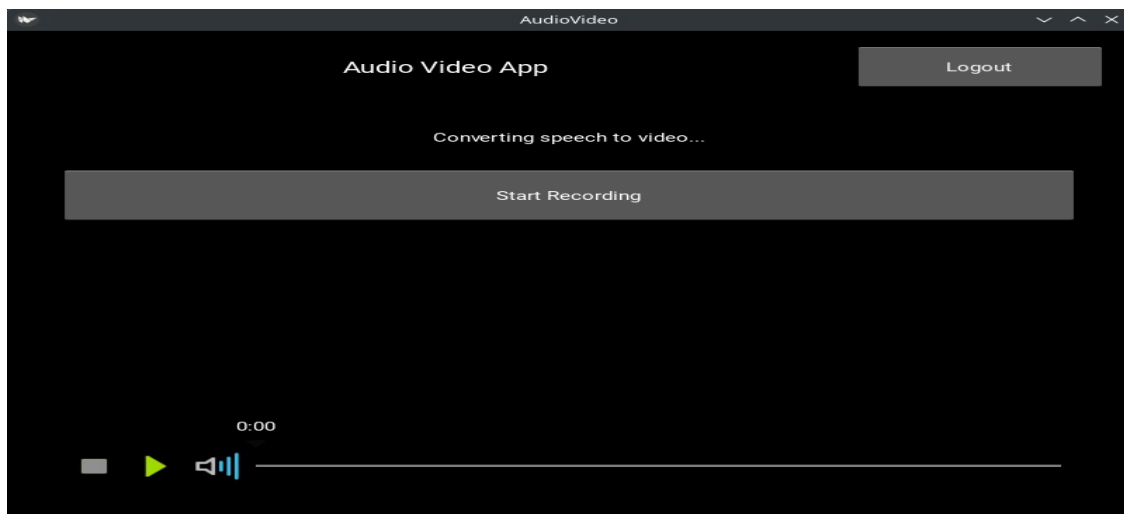


Fig 16. Converting Speech to Video

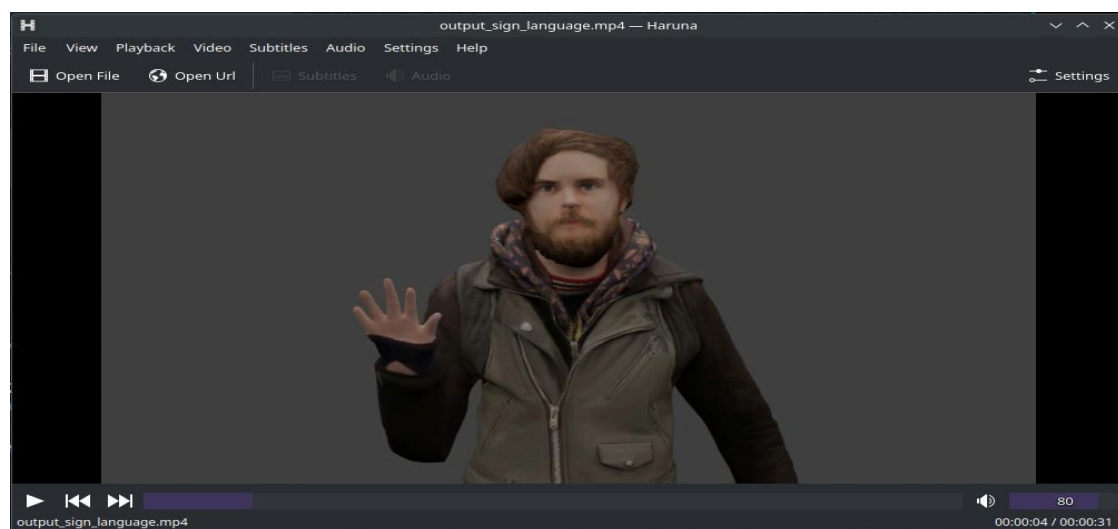


Fig 17. Final Video Output

## **6. Conclusion and Future Scope**

### **6.1 Conclusion**

The system proposed here for speech-to-sign-language conversion addresses the communication barriers faced by the hearing and speech-impaired communities with the world outside. Using best technologies such as Whisper for speech-to-text, state-of-the-art NLP models such as BERT and T5 for processing and keyword extraction, and animation tools such as Blender for sign language gesture generation, the system is well placed in efficiency and robustness. Bridging critical gaps such as multilinguality, noise reduction, real-time output, and different sign language support, this project aims to create an inclusive platform that would serve much better in real-world scenarios. The modular structure is also designed in a way to offer greater scalability, adaptability, and user-centricity, thus making it an important tool for accessibility and social inclusion.

### **6.2 Future Scope**

The future work of the project could possibly include a number of things. The first area could be further optimizing real time performance through combining any further hardware accelerators along with lightweight model architectures. Adding support for more languages and regional sign languages will extend the use of the system to more people around the world. Three, some personalization features could be added based on AI, such as personalization adapting the true gestures on demand of the user or any dialect requirements.

In addition, advancements in emotion detection and contextual understanding will allow the system to communicate, not just the content of speech through signs but also its sentiment. The experience might then be taken to new measures by integrating the AR, VR with such a system. This could also extend the system to mobile platforms and edge devices so that it becomes portable and can be available in various settings. Last but not least, collaboration with linguists and the hearing impairment community is essential in fine-tuning the gestures and improving cultural and linguistic accuracy in the system. In that continuous improvement, this project would have the power of revolution in breaking communication barriers and making a better, more inclusive world possible for countries and populations to come.

## 7. References

- [1] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv. /abs/1810.0480
- [2] Kapoor, P., Mukhopadhyay, R., Hegde, S. B., Namboodiri, V., & Jawahar, C. V. (2021). Towards Automatic Speech to Sign Language Generation. ArXiv. /abs/2106.12790
- [3] Sharma, Purushottam & Tulsian, Devesh & Verma, Chaman & Sharma, Pratibha & Nancy, Nancy. (2022). Translating Speech to Indian Sign Language Using Natural Language Processing. Future Internet. 14. 253. 10.3390/fi14090253.
- [4] Das Chakladar, D.; Kumar, P.; Mandal, S.; Roy, P.P.; Iwamura, M.; Kim, B.-G. 3D Avatar Approach for Continuous Sign Movement Using Speech/Text. Appl.Sci. 2021, 11, 3439. <https://doi.org/10.3390/app11083439>
- [5] Yash Verma, R. S. Anand, Gesture generation by Robotic hand for aiding Speech and hand of hearing persons based on Indian sign Language. 2024. <https://doi.org/10.1016/j.hellyon.2024.929678>
- [6] Saud Mian Qaisar, Sorah Niyazi, Abdulhanit Subasi, Efficient Isolated speech to sign conversion Based on Adaptive Rate Processing. <https://doi.org/10.1016/j.procs-2019-12.083>.
- [7] Jain, N., Goyal, M., Gupta, A., & Kumar, V. (2021). Speech to text conversion and sentiment analysis on speaker specific data. International Research Journal of Modernization in Engineering Technology and Science, 3(6), 3050-3057.
- [8] Nagdewani, S., & Jain, A. (2020). A review on methods for speech-to-text and text-to-speech conversion. International Research Journal of Engineering and Technology, 7(5), 4459-4471.
- [9] Stoll, S., Camgoz, N. C., Hadfield, S., & Bowden, R. (2020). Text2Sign: Towards Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks. International Journal of Computer Vision, 128(8), 891-908.
- [10] Prachi waghmare, Ashwini deshpane, Improvement of BERT-based Multi-Head Self-Attention Transformer for Translating Marathi text into Marathi sign language gross. <https://doi.org/10.1145/3687304> "https://doi.org/10.1145/3687304%20Accepted%2018%20July%202024" HYPERLINK "https://doi.org/10.1145/3687304%20Accepted%2018%20July%202024"Accepted 18 July 2024
- [11] Debashis Das Chakladar, Pradeep Kumar, Partha Roy, 3D Avatar Method for Continuous Sign Movement by Speech/Text <https://www.researchgate.net/publication/> April 2021
- [12] Mounika Kanakanti, A Multi Modal Approach to Speech-to-Sign Language Generation. mounika.k@research.iiit.ac.in May 2024
- [13] Othman, Achraf, and Mohamed Jemni. "Designing High Accuracy Statistical Machine Translation for Sign Language Using Parallel Corpus: Case Study English and American Sign Language." JITR vol.12, no.2 2019: pp.134-158. <http://doi.org/10.4018/JITR.2019040108>.
- [14] Sanaullah, M., Ahmad, B., Kashif, M., Safdar, T., Hassan, M. et al. (2022). A real-time automatic translation of text to sign language. *Computers, Materials & Continua*, 70(2), 2471-2488. <https://doi.org/10.32604/cmc.2022.019420>.

- [15] Rothwell, A., Moorkens, J., Fernández-Parra, M., Drugan, J., & Austermuehl, F. (2023). *Translation Tools and Technologies* (1st ed.). Routledge. <https://doi.org/10.4324/9781003160793>
- [16] Sethiya, N., & Maurya, C. K. (2025). End-to-End Speech-to-Text Translation: A Survey. *Computer Speech & Language*, 90, 101751. <https://doi.org/10.1016/j.csl.2024.101751>
- [17] J. Wu et al., "On Decoder-Only Architecture For Speech-to-Text and Large Language Model Integration," 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Taipei, Taiwan, 2023, pp. 1-8, doi: 10.1109/ASRU57964.2023.10389705.
- [18] T. N. Sainath et al., "JOIST: A Joint Speech and Text Streaming Model for ASR," 2022 IEEE Spoken Language Technology Workshop (SLT), Doha, Qatar, 2023, pp. 52-59, doi: 10.1109/SLT54892.2023.10022774.
- [19] Kheddar, H., Hemis, M., & Himeur, Y. (2024). Automatic speech recognition using advanced deep learning approaches: A survey. *Information Fusion*, 109, 102422. <https://doi.org/10.1016/j.inffus.2024.102422>
- [20] Zhang, Y., Han, W., Qin, J., Wang, Y., Bapna, A., Chen, Z., Chen, N., Li, B., Axelrod, V., Wang, G., Meng, Z., Hu, K., Rosenberg, A., Prabhavalkar, R., Park, D. S., Haghani, P., Riesa, J., Perng, G., Soltau, H., . . . Wu, Y. (2023). Google USM: Scaling Automatic Speech Recognition Beyond 100 Languages. *ArXiv*. <https://arxiv.org/abs/2303.01037>
- [21] Kheddar, H., Himeur, Y., Al-Maadeed, S., Amira, A., & Bensaali, F. (2023). Deep transfer learning for automatic speech recognition: Towards better generalization. *Knowledge-Based Systems*, 277, 110851. <https://doi.org/10.1016/j.knosys.2023.110851>





# 19% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




## Filtered from the Report

- Bibliography
- Small Matches (less than 8 words)
- Crossref database
- Crossref posted content database

## Match Groups

-  **165** Not Cited or Quoted 19%  
Matches with neither in-text citation nor quotation marks
-  **0** Missing Quotations 0%  
Matches that are still very similar to source material
-  **0** Missing Citation 0%  
Matches that have quotation marks, but no in-text citation
-  **0** Cited and Quoted 0%  
Matches with in-text citation present, but no quotation marks

## Top Sources

- 15%  Internet sources
- 4%  Publications
- 16%  Submitted works (Student Papers)

## Integrity Flags

### 0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.