

Audio To Sign Language Translator

Submitted in partial fulfillment of the requirements

of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

By

Juhi Gianani

Priya Gianchandani

Ankita Kar

Guide:

PROF. JAYANT GADGE

(Professor, Department of Computer Engineering, TSEC)



Computer Engineering Department
Thadomal Shahani Engineering College
University of Mumbai
2019-2020

Audio To Sign Language Translator

Submitted in partial fulfillment of the requirements

of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

By

Group No: 32

Roll No.	Name
1604031	Juhi Gianani
1604032	Priya Gianchandani
1604051	Ankita Kar

Guide:

PROF. JAYANT GADGE

(Associate Professor, Department of Computer Engineering, TSEC)



Computer Engineering Department
Thadomal Shahani Engineering College
University of Mumbai
2019-2020

CERTIFICATE

This is to certify that the project entitled “**Audio To Sign Language Translator**” is a bonafide work of

Roll No.	Name
1604031	Juhi Gianani
1604032	Priya Gianchandani
1604051	Ankita Kar

Submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of “**BACHELOR OF ENGINEERING**” in “**COMPUTER ENGINEERING**”

Prof. Jayant Gadge

Guide

Dr. Tanuja Sarode

Head of Department

Dr. G. T. Thampi

Principal

Project Report Approval for B.E.

Project report entitled *Audio To Sign Language Translator* by

Roll No.	Name
1604031	Juhi Gianani
1604032	Priya Gianchandani
1604051	Ankita Kar

is approved for the degree of ***“BACHELOR OF ENGINEERING” in
“COMPUTER ENGINEERING”***

Examiners:

1. _____

2. _____

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

1) _____

Juhi Gianani 1604031

2) _____

Priya Gianchandani 1604032

3) _____

Ankita Kar 1604051

Date:

Abstract

Speech is the main channel between people to communicate with each other. In the recent years, there has been rapid increase in the number of deaf and dumb victims due to birth defects, accidents and oral diseases. Since deaf and dumb people cannot communicate with normal person so they have to depend on some sort of visual communication. There are many languages spoken all around the world and interpreted. “Special people”, that is people who have difficulty in speaking and hearing “The dumb” and “The deaf” people respectively find it difficult to understand what exactly the other person is trying to express and so with the deaf people. Sometimes people interpret these messages wrongly either through sign language or through lip reading or lip sync. This project is made in such a way to help these specially challenged people hold equal par in the society.

Sign language is a language which uses visually transmitted sign patterns to convey meaning. It is the combination of hand shapes, orientation and movement of hands, arms or body, and facial expressions. Our system is capable of recognizing sign-language symbols can be used as a means of communication with hard of hearing people. Our paper proposes a system to help those people to communicate with normal people without sophisticated devices like power, data gloves and colored finger cap and etc. Our system is designed as getting voice input (from normal person) converted to text then it is matching with database for sign symbol to display (to hard hearing receiver). This system makes deaf/dumb people to communicate easily with normal speaking person.

TABLE OF CONTENTS

List of Figures		i
List of Tables		ii
Chapter 1	Introduction	1
	1.1 Introduction	1
	1.2 Aim and Objective	2
	1.3 Scope	2
Chapter 2	Review of Literature	4
	2.1 Domain Explanation	4
	2.1.1 Natural Language Processing	5
	2.1.2 Sign Languages	6
	2.1.3 Indian Sign Language	7
	2.2 Existing Solution	8
	2.2.1 Audio to Text Conversion	8
	2.2.2 Text to Sign Language Conversion	10
	2.3 Hardware and Software Requirements	10
	2.3.1 Hardware Requirements	11
	2.3.2 Software Requirements	11
Chapter 3	Analysis	12
	3.1 Functional Requirements	12
	3.2 Non-Functional Requirements	13
	3.3 Dataset	14
	3.4 Proposed System	16
	3.4.1 Audio to Text Converter	17
	3.4.2 Input Parser	18
	3.4.3 ISL Generator	18
	3.4.4 Video Generator	22

	3.5	Illustrations of Proposed System	23
	3.5.1	Illustration 1	23
	3.5.2	Illustration 2	24
Chapter 4		Design	26
	4.1	Design Consideration	26
	4.2	Design Details	27
	4.2.1	Use Case Diagram	27
	4.2.2	Sequence Diagram	28
	4.3	GUI Design	29
Chapter 5		Implementation and Results	32
	5.1	Implementation Details	32
	5.1.1	Audio to Text Converter	33
	5.1.2	Input Parser	34
	5.1.3	ISL Generator	35
	5.1.4	Video Generator	38
	5.2	Result and Evaluation	39
Chapter 6		Conclusion and Future Work	45
	6.1	Conclusion	45
	6.2	Future Work	45
References			47
Acknowledgement			48

List of Figures

Figure No.	Description	Page No
Figure 2.1	Alphabets and Numbers of American Sign Language	6
Figure 2.2	General Block Diagram of Audio/Video to Text Converter	9
Figure 3.1	Alphabet Chart of Indian Sign Language	14
Figure 3.2	Numeric Chart of Indian Sign Language	15
Figure 3.3	List of Words in Dataset	15
Figure 3.4	ISL Video Dataset	16
Figure 3.5	System Architecture	17
Figure 3.6	Example of Tokenization	18
Figure 3.7	Proposed System Block Diagram	22
Figure 3.8	Output of Illustration 1	24
Figure 3.9	Output of Illustration 2	25
Figure 4.1	Use Case Diagram	28
Figure 4.2	Sequence Diagram	29
Figure 4.3	Landing Page	30
Figure 4.4	Demo Section Part 1	30
Figure 4.5	Demo Section Part 2	31
Figure 4.6	About Section	31
Figure 5.1	Audio to Text Conversion	33
Figure 5.2	CoreNLPServer Implementation	34
Figure 5.3	CoreNLPParser Implementation	35
Figure 5.4	Lemmatizing Tokens	36
Figure 5.5	Filtering Stop Words	36
Figure 5.6	Dataset Words	37
Figure 5.7	Processing of ISL Sentence	37
Figure 5.8	Merging of Video Files	38
Figure 5.9	Console Output	40
Figure 5.10	Screenshots of Output	43

List of Tables

Table No.	Description	Page No
Table 3.1	Comparison of Grammar of English and ISL	19
Table 3.2	Examples of Grammatical Reordering of Words of English Sentence	20

Chapter 1

Introduction

This chapter gives an introduction about the project, problem definition, scope, aim and objective of the project.

1.1 Introduction

Technology is rapidly changing and improving the way the world operates. Barriers for people who are deaf are diminishing as projects of the past two decades have unfolded. Practically, world's hard hearing and hard speaking people has been in a difficult situation in the society because of their inability to communicate vocally with normal speaking and hearing people in connection with that the indifference of others to learn their language, the sign language. With the arrival of multimedia, animation and other computer technologies, it is now

becoming possible to bridge the communication gap between the hearing-impaired and normal person. Using sign language hard speaking and hearing people could communicate among them and with normal people.

As far as a deaf person is concerned, having access to a sign language is very important for their social, emotional and linguistic growth. Sign language should be recognized as the first language of deaf people and their education can be proceeded bilingually in the national sign language as well as national written or spoken language. Through the use of artificial intelligence, researchers are striving to develop hardware and software that will impact the way deaf individuals communicate and learn.

1.2 Aim & Objective

The aim of this project is to design a convenient system that is helpful for the people who have hearing difficulties and in general who use very simple and effective method; sign language. Sign language is the main communication medium of the hearing impaired, in terms of automatic recognition. The ultimate aim should be to have a system that translates audio signals to speech. The objective of the project are as follows:

- Designing a close to real time system that performs audio to speech translation
- Designing various modules of the system that is required to complete the given task.
 - Audio Recognition module
 - Audio to speech conversion module
 - Usage of language models to solve ambiguities in speech recognition step

1.3 Scope

This system can translate any spoken language into sign language. For this, language models must be trained, which would be projects on their own. Since deaf people are usually

deprived of normal communication with other people, they have to rely on an interpreter or some visual communication. Now the interpreter cannot be available always, so this project can help eliminate the dependency on the interpreter. The system can be extended to incorporate the knowledge of facial expressions and body language too so that there is a complete understanding of the context and tone of the input speech. A mobile and web based version of the application will increase the reach to more people. Integrating hand gesture recognition system using computer vision for establishing 2-way communication system. This system can be used in schools, hospitals and public meetings, in the place of an interpreter, to convey message to the deaf. It can also be developed for different regional languages, in India, to solve the problem of communication with the deaf even for small regional language groups.

Chapter 2

Review of Literature

This chapter includes the domain explanation of the project and different existing solutions. The domain section contains a brief detail of the concepts used in the project. While the existing solution includes related solutions for this project.

2.1 Domain Explanation

This section gives detailed explanation about Natural Language Processing, Sign Languages and Indian Sign Language (ISL). Before proceeding with the existing solutions and design of the system, we should have a clear understanding about the following concepts.

2.1.1 Natural Language Processing

Natural language refers to the language spoken by human beings. There are around 6,500 spoken languages in the world and all of them are natural languages. Natural languages can take different forms such as written text, speech or signing. Natural Language Processing (NLP) is a branch of linguistics and artificial intelligence that helps computers to understand, interpret and manipulate human languages. In the past 20 years, there is a significant growth in the volume and variety of data that is being produced everyday and almost 80% of the data is unstructured text data. Text analytics, through the use of NLP is a very efficient way to extract the information in the unstructured text data and unlock its business value.

The ultimate goal of NLP is for computers to achieve human-like comprehension of texts or languages. The computer systems should be able to understand, draw references from, summarize, translate and generate accurate and natural human language text. Upon feeding enough data and training a model properly, computer systems can distinguish and try categorizing various parts of speech (noun, verb, adjective, etc.) based on previously fed data and experiences. There are in general five stages of NLP. They are as follows:

1. Lexical Analysis:

It deals with analyzing the structure of each word in the sentence. In this stage the entire text is divided into paragraphs, sentences and then words. This level of linguistic processing utilizes a language's lexicon, which is collection of individual lexemes.

2. Syntactic Analysis:

This stage involves analysis of words in the sentence for grammar and arranging words in a manner that shows relationship among them. Linear sequences of words are transformed into structure that show how the words relate to each other.

3. Semantic Analysis:

In this stage, the exact dictionary meaning of the words is drawn out. It concerns with what words mean and how these words are combined to form the meaningful sentence. A transformation is made from the input text to an internal representation which reflects the meaning.

4. Discourse Analysis:

This stage concerns with how the immediately preceding sentence affect the interpretation of the next sentence. The meaning of a sentence depends in the meaning of the sentence just before it. In addition, it also brings about the meaning of the immediately next sentence.

5. Pragmatic Analysis:

This stage concerns how sentences are used in different situations and how it affects the interpretation of the sentence. During this stage, what was said is reinterpreted as what it actually meant. It involves deriving those aspects of the language which require real world knowledge.

2.1.2 Sign Languages

A sign language is a natural visual-spatial language that uses three-dimensional space to articulate linguistic utterances instead of sound to convey meaning. It involves simultaneous usage of different orientation and movement of hands, arms, upper body and facial expressions to express the linguistic message. Hard speaking and hard hearing people communicate among themselves and with normal people through sign language. It is not only used by deaf and dumb people but by those who suffer with Autism Spectrum Disorder (ASD).

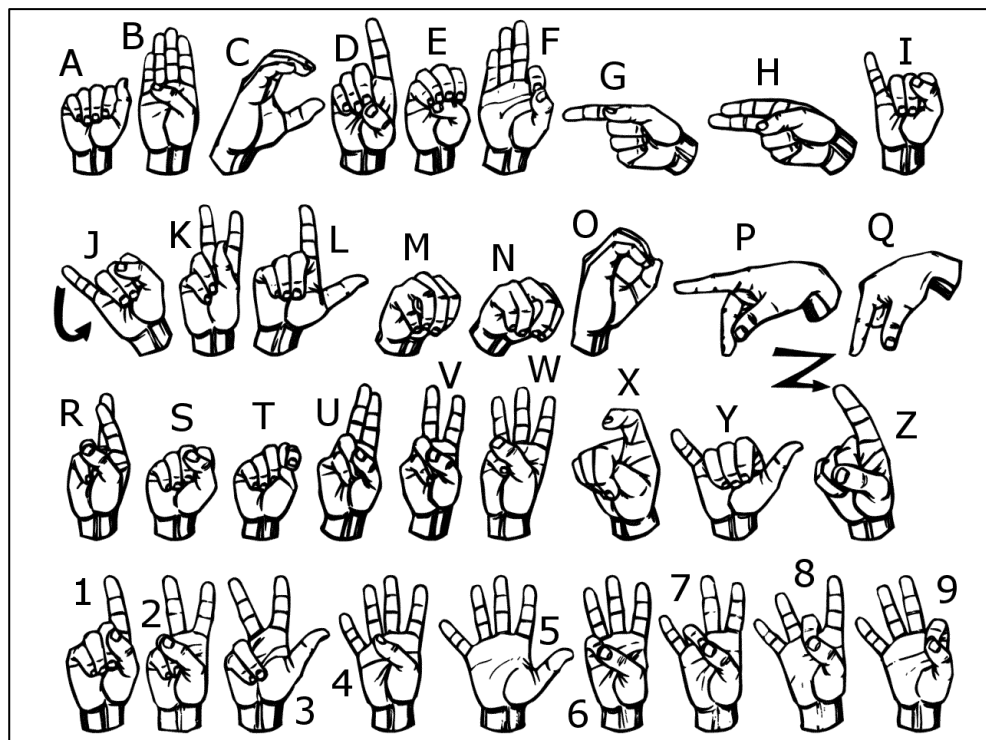


Figure 2.1 Alphabets and Numbers of American Sign Language

Just like there are many spoken languages in the world like English, French, Hindi, etc., similarly there are different sign languages and different expressions used by hearing disabled people worldwide. The sign language used in USA is American Sign Language (ASL), British Sign Language (BSL) is used in Britain and Indian Sign Language (ISL) is used in India. Some of the other types of sign languages as Brazilian Sign Language, Chinese Sign Language, French Sign Language, etc.

2.1.3 Indian Sign Language

Indian Sign Language (ISL), also known as Indo-Pakistan Sign Language (IPSL), is the predominant sign language in South Asia. While the sign system in ISL appears to be indigenous, elements in ISL are derived from British Sign Language (BSL). Despite the common assumption that ISL is manual representation of spoken Hindi or English language, ISL is unrelated to either language and has its own grammar. Particularly, research on ISL grammar started in 1978 and it has been found that ISL is a complete natural language with its own grammar and syntax. There are three aspects of ISL, lexicon, syntax, and spatial grammar.

The Indian Sign Language is not the same as the manual representation of spoken English or Hindi language. ISL consists of various non-manual gestures including mouth patterns, mouth gestures, facial expressions, body position and eye gaze. ISL has essentially a Subject-Object-Verb word order which is unlike English which is Subject-Verb-Object. It has certain unique distinct features like

1. All the sign representation for numbers are done with appropriate hand gestures for every number. For example the sign for 45 will be the representation of 4 followed by sign representation of 5.
2. The sign for family relationships are preceded by signs for ‘male/man’ and ‘female/woman’.
3. The interrogative sentences having words like WHAT, WHERE etc. are represented by placing these questions at the end of the sentences.

It has been estimated that there are between 0.9 and 14 million hearing impaired in India and perhaps “one of every five people who are deaf in the world, lives in India”, making it the

country with largest number of Deaf, and perhaps also the largest number of sign language users. With the arrival of multimedia, animation and other computer technologies, it is now becoming possible to bridge the communication gap between the hearing-impaired and normal person. Sign language is a visual language that is used by deaf people as their mother tongue. According to census 2011, 76-89% of Indian Deaf have no knowledge about language, either signed or spoken. The reason behind this low literacy can be lack of sign language interpreters, unavailability of ISL tools and lack of research on ISL.

2.2 Existing Solution

There are existing solutions for two sub problems of this problem definition. There are existing solutions for Audio to Text Conversion and Text to Sign Conversion. The solutions for these two sub problems are discussed below.

2.2.1 Audio to Text Conversion

Speech recognition is the algorithm that allows recognition of the speech and translates into some purposes. Speech recognition is designed to translate voice into text. The research of speech recognition is conducted by applying audio recognition and video speech information process. The general audio-visual speech recognition is divided into two parallel stages: the lips reading and the voice recognition. The general flow of the audio-visual speech recognition is shown below. In this system, voice and image are divided, and then they are processed separately. Voice input passes through Acoustic Process and Prob Calculation blocks. Lip image passes through Image Process and Prob Calculation block. After passing all processes in each part, integration block combines result of two inputs, and it is finally processed into recognition part, in this diagram, search block is the final process.

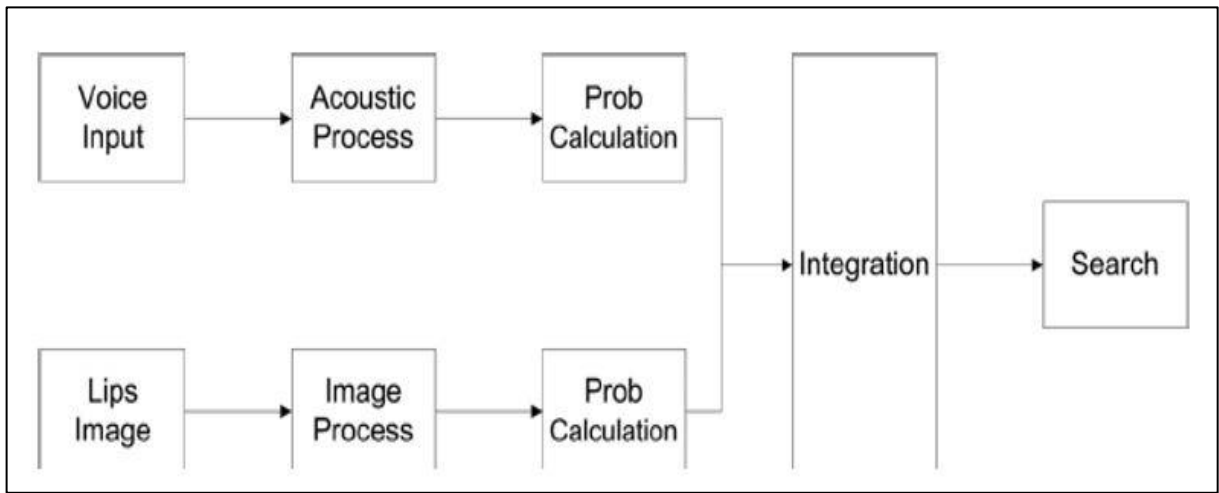


Figure 2.2 General Block Diagram of Audio/Video to Text Converter

1. A Review on Speech To Text Conversion Methods, International Journal of Advanced Research in Computer Engineering and Technology, Miss Prachi Khilari, Prof. Bhope V.P.

- Input: Audio voice
- Output: Text Language
- Process:
 - i. This system has two phases. The first phase is the Training Phase during which the system learns the references patterns representing different speech sounds that constitute the vocabulary of the application.
 - ii. Each reference is learned from the spoken example and stored in the form of templates obtained by some averaging method or models that characterize the statistical properties of patterns.
 - iii. The second phase is the Recognizing Phase in which unknown input pattern is identified by considering the set of references.

2. Voice Recognition System: Speech To Text, Journal of Applied and Fundamental Sciences, Prerana Das, Kakalai Acharjee, Pranab Das and Vijay Prasad.

- Input: Voice input from user
- Output: Text Output
- Process:
 - i. Feature Extraction will be done on the input voice using Vector Quantization.
 - ii. After extraction, it will be decoded using Acoustic Model and Language Model.
 - iii. These models will be created using Hidden Markov Model (HMM).
 - iv. The desired output will be shown on Matlab interface

2.2.2 Text to Sign Language Conversion

This process is mostly done using Natural Language Processing and Neural Network. Different types of Neural Networks are used to implement Natural Language Processing. Some of the solutions are as follows.

1. Sign Language Converter, International Journal of Computer Science and Engineering Survey, Taner Arsan, Volume 6, Number 2 August 2015

- Input: Text in sequence of letter format
- Output: Sign of each letter
- Process:
 - i. The input text is matched with the letters in the dataset.
 - ii. The matching is done using confidence creation which is done using probability concept.
 - iii. The one which matches with one of the sign is displayed on the screen

Thus by incorporating a microphone, speech-to-text software, and a written language-to-sign-language translation component into a handheld computer, one could produce a conversational interpreting tool to provide real-time interpreting services for deaf signers in contexts where hiring a live interpreter would be impossible.

2.3 Hardware and Software Requirements

The following section covers hardware requirements and software requirements of this project. Since this project is deployed as a web application, it can run on desktop as well as on mobile. It can run on any operating system of desktop and mobile.

2.3.1 Hardware Requirements

The audio input to the system will be given through a microphone. The microphone should be of good quality which can fetch the audio input quickly. It should have a minimum frequency of 20 Hz and Sound to Noise ratio should be more than 50.

2.3.2 Software Requirements

Since this project is deployed as a website, it can be viewed on desktop as well as in mobile phone. In case of desktop, the operating system should be Microsoft Windows 7 or higher versions of Microsoft Windows. For mobile phones, Android version should be Lollipop (API Level 21) or higher. For viewing the website, any type of browser can be used.

Chapter 3

Analysis

This chapter includes details of functional requirements, non-functional requirements, dataset of the project, proposed system of the project and illustrations of the proposed system.

3.1 Functional Requirements

Functional Requirements specifies a function that a system component must be able to perform. It can be documented in various ways. A functional requirement is used to help the reader understand why the requirement is needed, and to track the requirement through the development of the system. Functional requirements maybe calculations, technical details, data manipulation and processing and other specific functionality that define what a system is

supposed to accomplish. The most common ones are written descriptions in documents and use cases. The functional requirements of this project are:

1. The system should be able to take input in the form of audio through microphone.
2. The system should be able to perform conversion of voice input to Indian Sign Language implicitly.
3. Appropriate ISL should be display for corresponding voice input.

3.2 Non Functional Requirements

Non functional requirements specify the criteria used to judge the operation of the system, rather than the specific behaviors. Non-Functional requirements are a description of features, characteristics and attributes of the system as well as any constraint that may limit the boundaries of the proposed system. The Non-Functional requirements are essentially based on the performance, information, economy, control and security efficiency and services. Based on this, the non-functional requirements are as follows

1. Performance Requirement:

The system should be able to convert spoken words to ISL quickly. For desired performance, fetching audio and processing it, converting it to ISL, response time, processing speed of the system must be considered.

2. Reliability:

The solution should provide reliability to the user that the system will run all the features mentioned in this document are perfectly available and executing perfectly. It should be tested and debugged completely. All exceptions should be well handled.

3. Accuracy:

The solution should be able to reach the desired level of accuracy.

4. Usability:

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

5. Portability:

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

3.3 Dataset

The dataset contains images of signs of alphabets (A to Z) in Indian Sign Language. These individual images of each alphabet character are further processed and converted to video. Along with these videos, the dataset also consists of 216 videos of different words. Each video is in (.mp4) format and the name of the video is saved as the word in lowercase. For example, if the dataset has a video of the word “Apple” then it will be saved as “apple.mp4”. The images dataset that were further used and processed in order to get the video format are shown below.

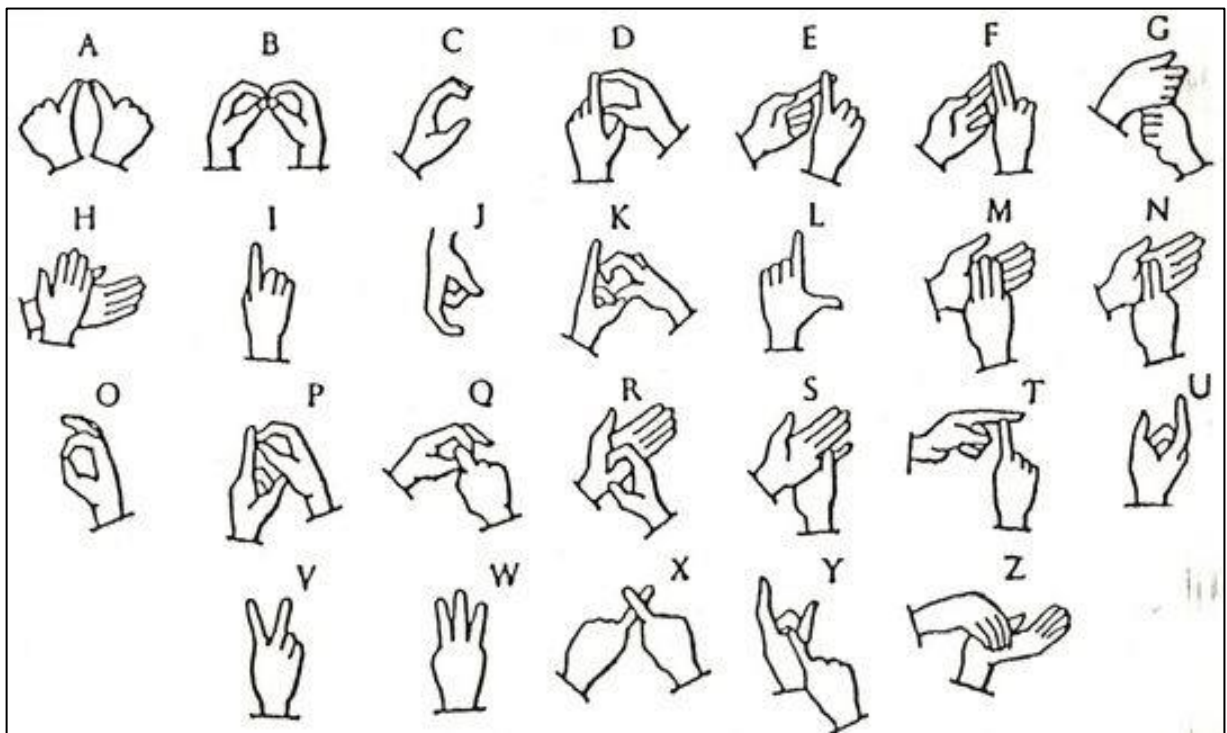


Figure 3.1 Alphabet Chart of Indian Sign Language

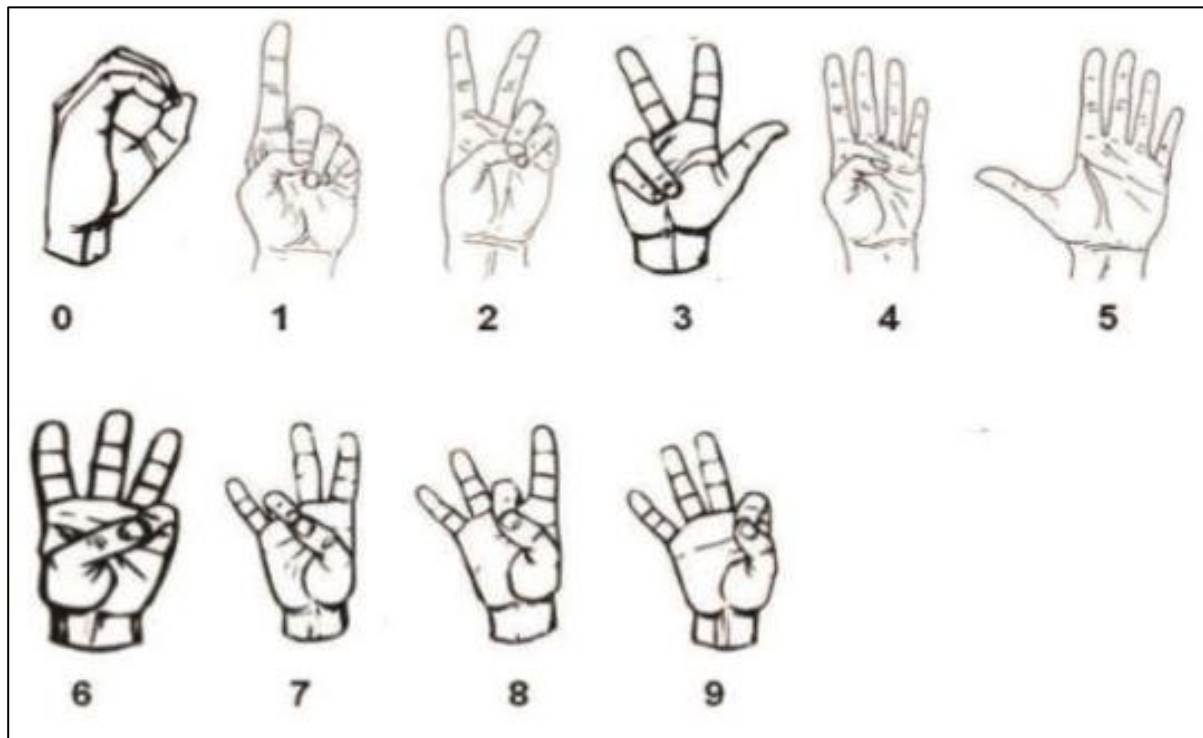


Figure 3.2 Numeric Chart of Indian Sign Language

The following images show the list of words whose video is available in our dataset and their corresponding videos in a folder.

```
[ 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
  'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
  'about', 'above', 'across', 'actress', 'address', 'aeroplane', 'after', 'again',
  'alive', 'allergy', 'aluminium', 'always', 'apple', 'area', 'around', 'assembly',
  'baby', 'badminton', 'ball', 'bandh', 'basketball', 'bat', 'before', 'behind',
  'bell', 'below', 'bengali', 'between', 'birth', 'bloodpressure', 'boat', 'bogie',
  'boxing', 'boy', 'brass', 'brother', 'cancer', 'capital', 'carrom', 'cement',
  'centimeter', 'chess', 'cholera', 'christmas', 'coal', 'coconut', 'cold', 'corn',
  'cough', 'cremate', 'cure', 'diabetes', 'die', 'disease', 'down', 'durga', 'during',
  'election', 'engine', 'exercise', 'far', 'flag', 'foot', 'friend', 'from', 'fruit',
  'funeral', 'game', 'ghost', 'government', 'grain', 'gram', 'grandfather',
  'grandmother', 'grandson', 'groundnut', 'guard', 'gujarati', 'health', 'helicopter',
  'here', 'hindi', 'history', 'hockey', 'holi', 'housefly', 'inch', 'jackfruit',
  'jaundice', 'karate', 'kilogram', 'kilometer', 'king', 'kite', 'leader', 'leather',
  'lemon', 'leprosy', 'lighthouse', 'litre', 'lose', 'magnet', 'malayalam', 'marathi',
  'mary', 'measles', 'measure', 'measurement', 'meter', 'mile', 'mililitre', 'minister',
  'mug', 'mums', 'mumps', 'near', 'neighbour', 'newspaper', 'olympics', 'out', 'over',
  'parliament', 'plastic', 'poison', 'pomegranate', 'postman', 'power', 'pray',
  'president', 'priest', 'queen', 'race', 'railway', 'relative', 'reservation', 'rice',
  'riots', 'rocket', 'room', 'rubber', 'run', 'scale', 'seat', 'ship', 'sick', 'silk',
  'silver', 'sometime', 'son', 'steel', 'strike', 'surname', 'tamil', 'telgu',
  'temperature', 'there', 'through', 'ticket', 'toilet', 'towards', 'trophy', 'twin',
  'under', 'urdu', 'vote', 'wedding', 'weight', 'wheat', 'wife', 'win', 'with', 'without',
  'woman', 'wood', 'wool', 'wound' ]
```

Figure 3.3 List of Words in Dataset



Figure 3.4 ISL Video Dataset

3.4 Proposed System

The proposed system takes audio as input, converts the audio to English text, which is further translated to ISL grammar by using parse tree structure. Machine based translation has been used to translate English to ISL based grammar. The parse tree of English input is converted to parse tree of ISL grammar sentence. The modified parse tree is the converted to text which is then processed and the final video of the ISL signs is displayed. The system architecture consists of the following modules:

1. Audio to Text Converter
2. Input Parser
3. ISL Generator
4. Video Generator

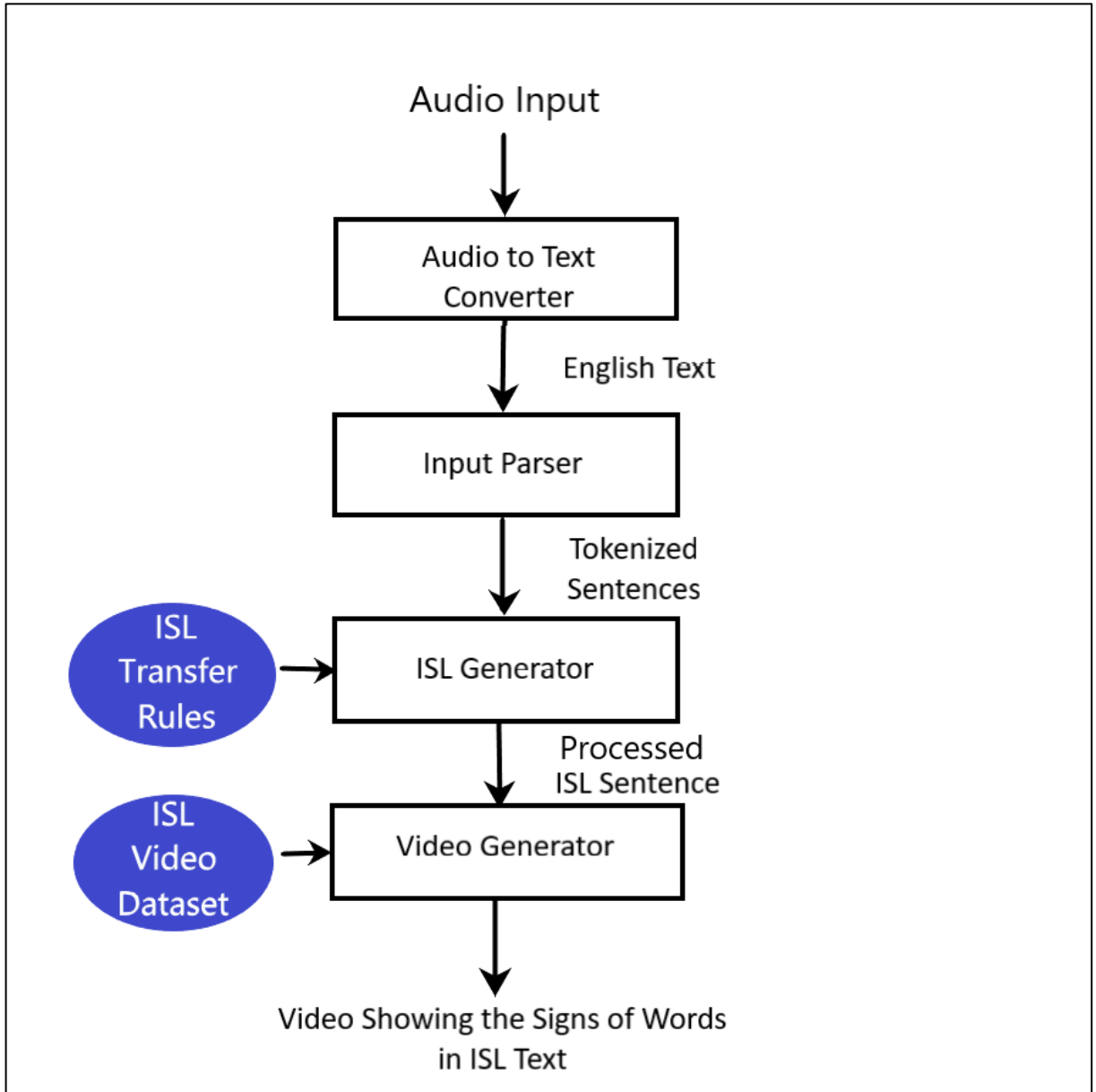


Figure 3.5 System Architecture

3.4.1 Audio to Text Converter

The normal speaking person gives the audio input through the microphone of mobile phone or computer. The audio is processed for noise removal. After noise removal, the audio is converted to text using HMM algorithm. In this project we will use Google Speech API for converting speech to text which internally uses HMM algorithm. The output of this module will be an English text string which will be passed to Input Parser module for further processing.

3.4.2 Input Parser

The input text is tokenized into sentences using Natural Language Tools. Tokenization is the task of breaking the text into pieces called as tokens and at the same time eliminating certain characters such as punctuations. A tokenizer divides text into a sequence of tokens which roughly corresponds to “words”. The parser creates a phrase structure tree of the input text which contains all the tokens. The tree is passed to the ISL Generator module.

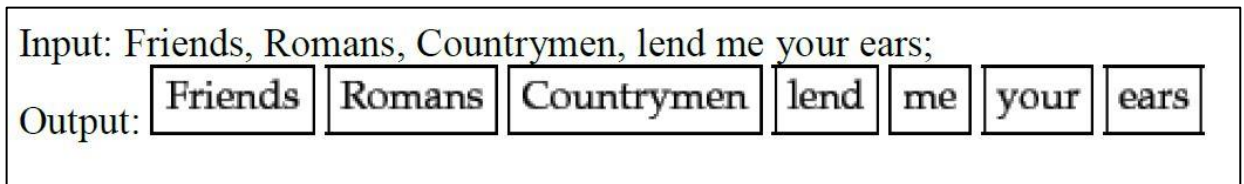


Figure 3.6 Example of Tokenization

3.4.3 ISL Generator

In this module, the English tokens are rearranged to form the ISL sentence. We will use ISL grammar rules to modify the phrase structure tree, which we got from the previous module, such that the modified tree now represents the structure and grammar of ISL. Translation of one language to another spoken language is a complex task if both languages have different grammar rules. The complexity increases many folds when source language is spoken language and the target language is a sign language. For translation English text to Indian Sign Language, we must first have a look at the comparison of grammar of both the languages.

English Language Grammar	Indian Sign Language Grammar
English grammar is well structure and a lot of research work has been carried out to define the rules for it. English grammar follows subject- verb-object order.	ISL is invented by deaf and a little work has been done to study the grammar of this language. The structure of sentences in ISL follow subject-object-verb order.
English language uses various forms of verbs and adjectives depending upon the type of sentences.	ISL does not use any inflections (gerund, suffixes or other forms). It uses the root form of the word.
English language has much larger dictionary.	ISL has a very limited dictionary, approximately 1800 words.
Question word in interrogative sentences is at the start of the word.	In ISL the question word is at the end of the sentence.
A lot of helping verbs, articles and conjunctions are used in the sentences of English.	In ISL, no articles, conjunctions or linking verbs are used.

Table 3.1 Comparison of Grammar of English and Indian Sign Language

In the phrase structure tree which is the input to this module, the noun phrase and the prepositional phrase are freezed but if there any verb phrase present in the tree, then it is checked recursively because the verb phrase may further be composed of noun phrase, prepositional phrase, verb phrase or even the sentence. For conversion of English sentence to a sentence as per ISL grammar rules, all the verb patterns are studies and rules are formed to convert English sentence int ISL sentence. Some of the rules for conversion are given below.

Verb Pattern	Rule	Input Sentence	Parsed Sentence	Output Sentence
verb + object	VP NP	go school	(VP (VB go) (NP (NN school)))	school go
subject + verb	NP VP	birds fly	(NP (NNS birds)) (VP (VBP fly))	birds fly
subject + verb + subject complement	NP VP NP	his brother became a soldier	(NP (PRPS his) (NN brother)) (VP (VBD became) (NP (DT a) (NN soldier)))	his brother a soldier became
subject + verb + indirect object + direct object	NP VP NP NP	i lent her my pen	(NP (FW i)) (VP (VBD lent) (NP (PRP her)) (NP (PRPS my) (NN pen)))	i her my pen lent
subject + verb	subject + verb	show me your hands	(VP (VBP show) (NP (PRP me))) (NP (PRPS your) (NNS hands))	me your hands show
subject + verb + direct object + preposition + prepositional object	NP VP NP PP	she made coffee for all of us	(NP (PRP she)) (VP (VBD made) (NP (NN coffee)) (PP (IN for) (NP (NP (DT all)) (PP (IN of) (NP (PRP us))))))	she coffee for all of us made
subject + verb + indirect object + direct object	VP NP PP	show your hands to me	(VP (VB show) (NP (PRPS your) (NNS hands)) (PP (TO to) (NP (PRP me))))	your hands to me show
subject + verb + preposition + prepositional object	NP VP NP	we are waiting for suresh	(NP (PRP we)) (VP (VBP are) (VP (VBG waiting) (PP (IN for) (NP (NN suresh)))))	we for suresh are waiting

Table 3.2 Examples of Grammatical Reordering or Words of English Sentence

In this module there are four phases. They are as follows

1. Modify SVO Structure to SOV Structure:

In this phase, the subject-verb-object (SVO) structure of the English sentence is converted to subject-object-verb (SOV) using the rules mentioned in Table 3.2. The phrase structure tree will be modified in this step.

2. Handling WH Interrogatives:

WH-interrogative markers like who, what, when and why always occur at the end of the sentence in Indian Sign Language. In this phase, we place the WH-interrogatives at the end by deleting node containing WH-interrogative words in the phrase structure tree and appending this node as the child of the last node.

3. Lemmatization:

In ISL, the word endings/suffixes or words ending with gerunds are never used. Every word in ISL is present in its root form. For grammatical reasons, sentences in English language can contain different forms of a word, such as organize, organizes, organizing. These forms of a word have to be converted to the root word. This can be done by stemming or lemmatization.

Stemming usually refers to a crude heuristic process that chops of the ends of words in hope of achieving the root word correctly and often includes the removal of derivation affixes. Lemmatization refers doing things with use of vocabulary and morphological analysis of words, aiming to remove inflectional endings and give the dictionary form of a word, which is called lemma. For example, if we take word “*saw*”, stemming might return just “*s*”, but lemmatization would attempt to return either “*see*” or “*saw*” depending on whether the token is used as a verb or a noun.

In our project we have lemmatized every token and in order to increase the accuracy, we have also assigned part-of-speech tagging to every word. This improves the accuracy of word to root word conversion.

4. Elimination:

According to rules of ISL, am/is/are/was/were i.e. linking verbs and articles like a, an, the, some are not used in ISL sentences. Such words are called “stop words”. These are very common words in English language but they have no meaning in Indian Sign Language and hence they have to be eliminated. In this step, we eliminate these stop words from the recorded tokens.

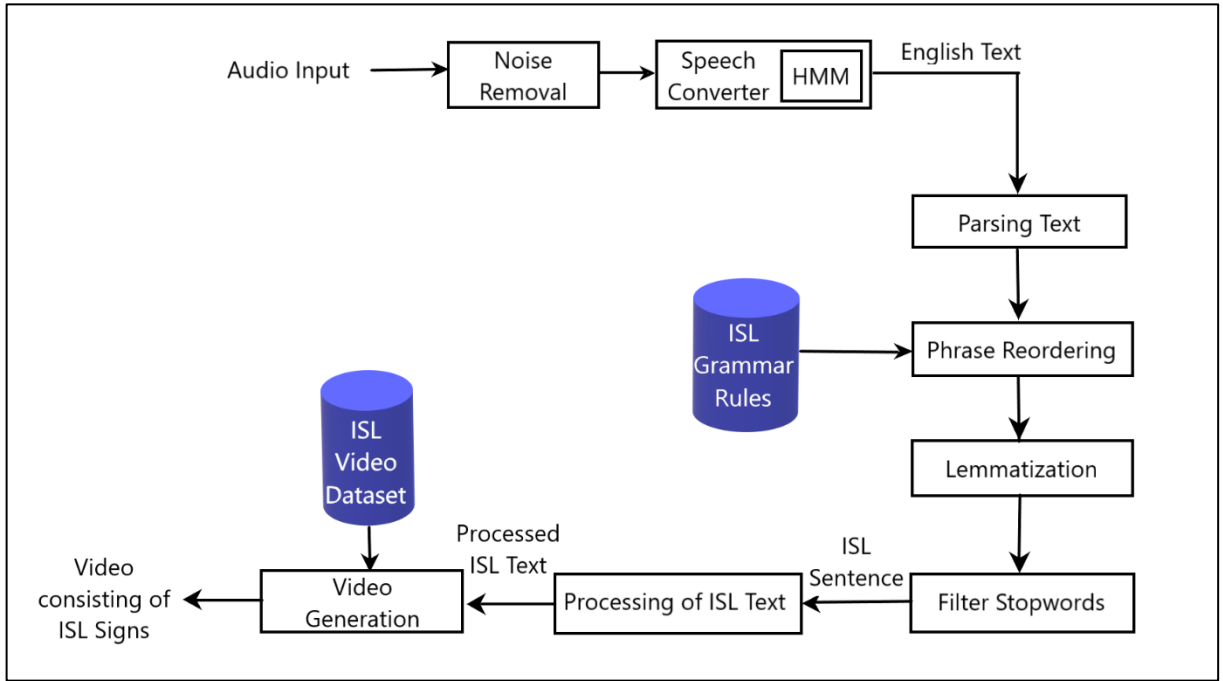


Figure 3.7 Proposed System Block Diagram

After these four phases, the text sentence in Indian Sign Language is achieved. For every word in the ISL sentence, we will check whether we have a video of ISL sign of that word available in our ISL video dataset. If the video is available, then we will keep the word as it is in the ISL text. But if we don't have the video available in the dataset, then we will break the word into letters.

For example, suppose the ISL sentence is "*i apple ate*". We have video of "*apple*" available in our dataset, but we don't have video of "*ate*". Hence the preprocessed ISL text will be "*i apple a t e*".

This way a processed ISL text is formed which is passed to the next module for generating the final output.

3.4.4 Video Generation

The main function of this module is to create the video of the ISL signs. For every word or letter in the processed ISL string, which is received from previous module, the system will find the video of the same word or letter in the ISL video dataset. After all the videos are found, they

are merged to form the final video of the ISL sentence. This video is then displayed to the hard speaking or hard listening person.

3.5 Illustration of Proposed System

This section contains two examples of how the audio input is taken into the system and how it is processed to generate the sign language output.

3.5.1 Illustration 1

1. Audio Input: - “Hello Everyone”

Once the audio input is given to the system, it is processed for noise removal and then it is passed to the Speech Converter. The Speech Converter module generates the English text of the audio input and then passes it to ISL generator module. The ISL generator creates the ISL sentence and processed ISL sentence.

2. ISL Processed Output: - {‘h’, ‘e’, ‘l’, ‘l’, ‘o’, ‘e’, ‘v’, ‘e’, ‘r’, ‘y’, ‘o’, ‘n’, ‘e’}

The processed ISL sentence is used for creating the video output. Here, the video of “hello” and “everyone” is not available in the ISL video dataset, hence the videos of all the letters will be merged and final output will be formed.

3. Final Output: -

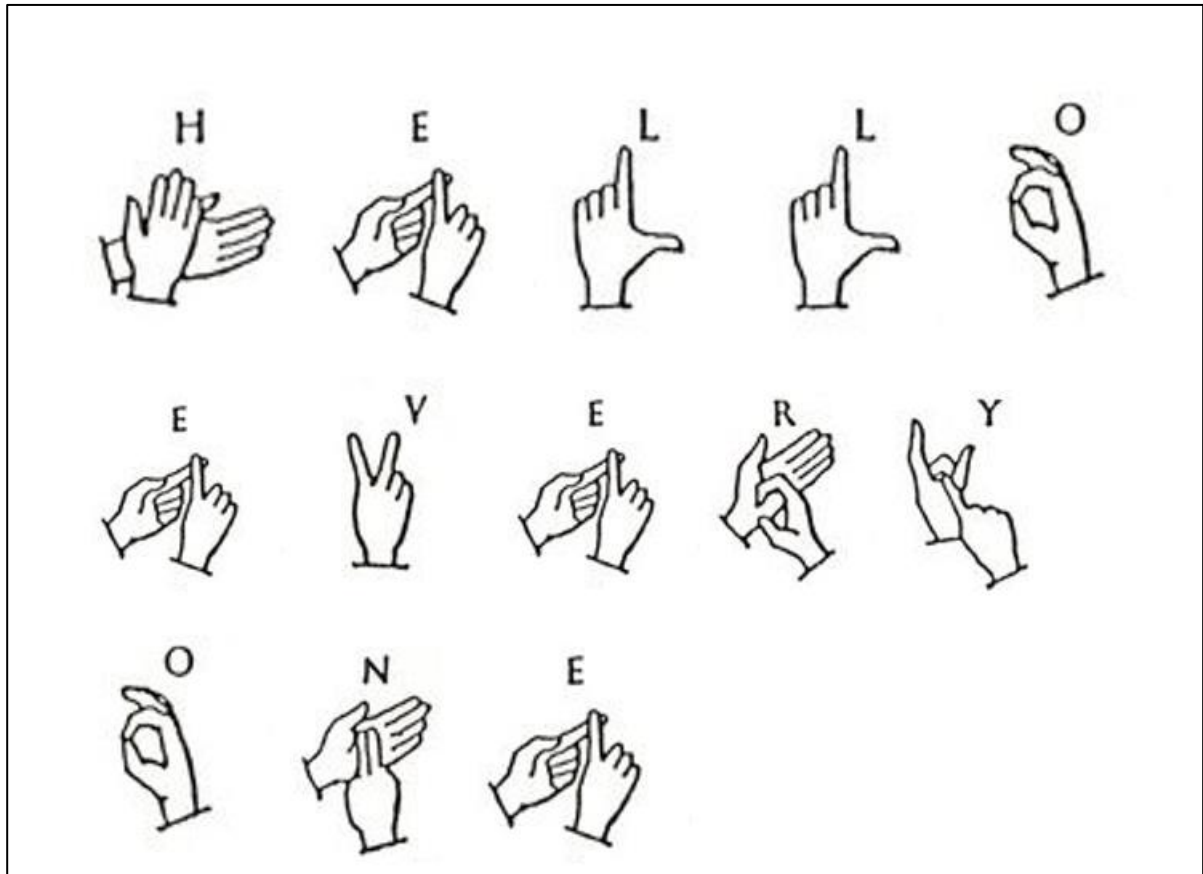


Figure 3.8 Output of Illustration 1

3.5.2 Illustration 2

1. Audio Input: - “Apple is red”

Once the audio input is given to the system, it is processed for noise removal and then it is passed to the Speech Converter. The Speech Converter module generates the English text of the audio input and then passes it to ISL generator module. The ISL generator creates the ISL sentence and processed ISL sentence.

2. ISL Preprocessed Output: - {'a', 'p', 'p', 'l', 'e', 'i', 's', 'r', 'e', 'd'}

The processed ISL sentence is used for creating the video output. Here, the video of “apple”, “is” and “red” is not available in the ISL video dataset, hence the videos of all the letters will be merged and final output will be formed.

3. Final Output: -

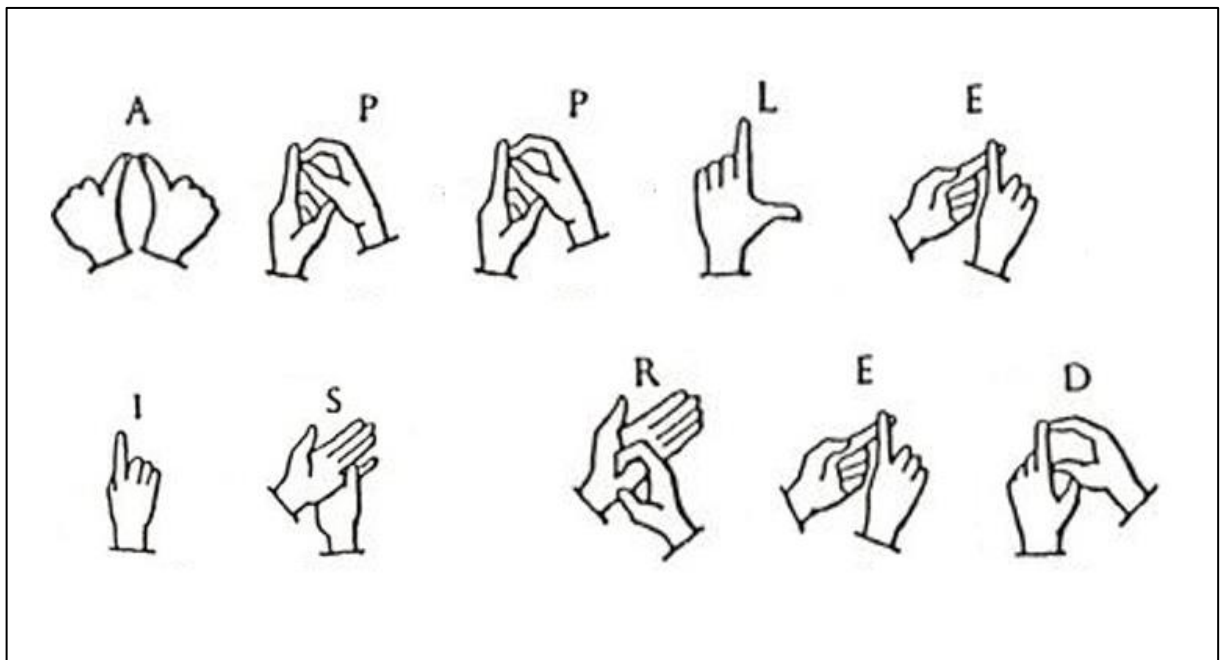


Figure 3.9 Output of Illustration 2

Chapter 4

Design

This chapter includes detailed explanation of design consideration, design details and GUI design of the project. This chapter gives a clear idea about the working of the project.

4.1 Design Consideration

The following factors were considered while designing the system.

1. Accessibility:

The system has been designed very user friendly so that it is accessible by user of any age group without any problem.

2. Usability:

The developed system is a website, hence it can be run on desktop as well as cellular phones. It is independent of the operating system of the device.

3. Operability:

The system doesn't require much maintenance that further saves manpower.

4. Extensibility:

The system can be further extended for different types of sign languages like American Sign Language, British Sign Language, etc.

4.2 Design Details

This section contains the use case diagram and the sequence diagram of the system. The use case diagram shows what exactly the system is doing and the sequence diagram describes the flow of processing of the system.

4.2.1 Use Case Diagram

The following use case diagram depicts the functionality of the system. It consists of two actors namely User and Impaired User. Both the actors communicate through a particular use case. User actor can only give input to the system via the Audio Input. Other use cases like the noise removal, convert to English text, etc. are all dependent on the previous use case. The second actor Impaired User will then receive the output in the form of video consisting of ISL signs.

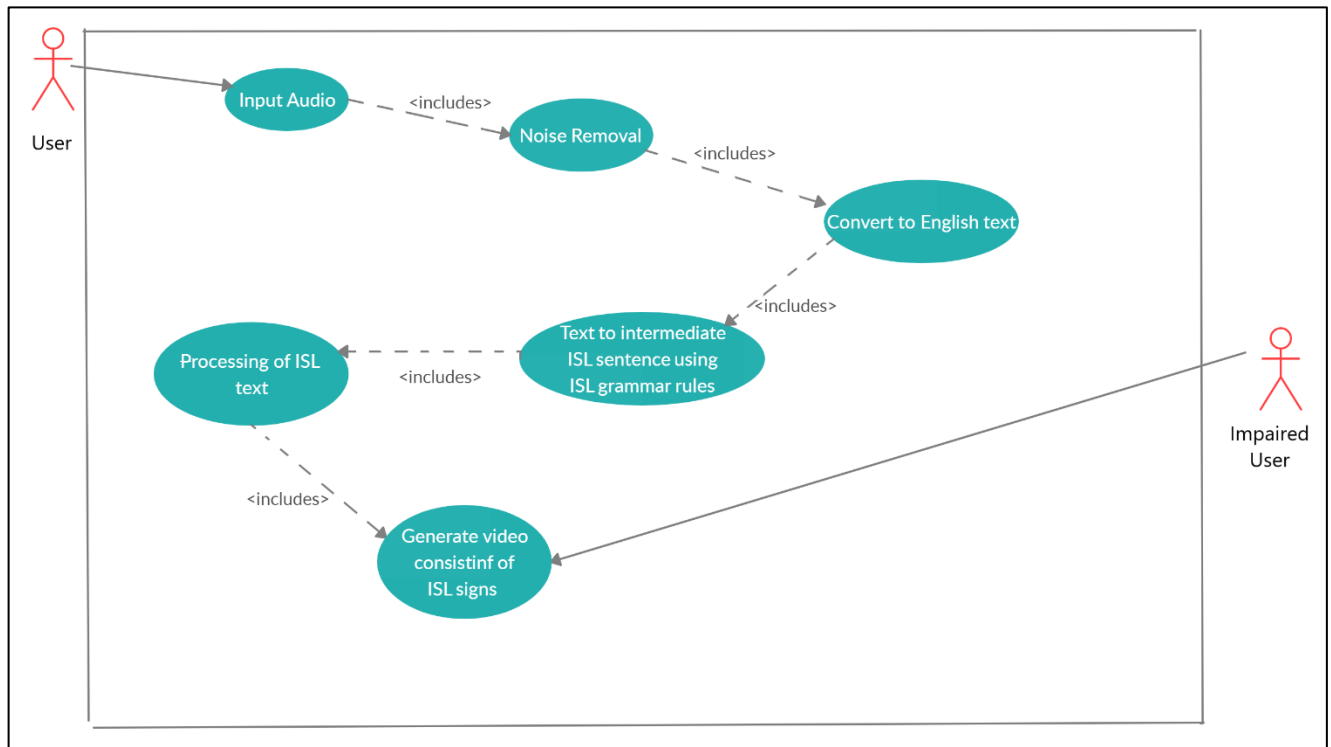


Figure 4.1 Use Case Diagram

4.2.2 Sequence Diagram

The User will give the audio input through Microphone. The Audio to Text Converter will convert the audio into English text. This English text will then be parsed by using the Input Parser and will be converted to tokenized sentence. The ISL generator will generate a processed ISL sentence using the tokenized English sentence. The English tokenized sentence will undergo phrase reordering, lemmatization, filtration of stop words, etc. This processed ISL sentence will then be given to the Video Generator which in turn produces the video showing the appropriate signs of words in ISL text.

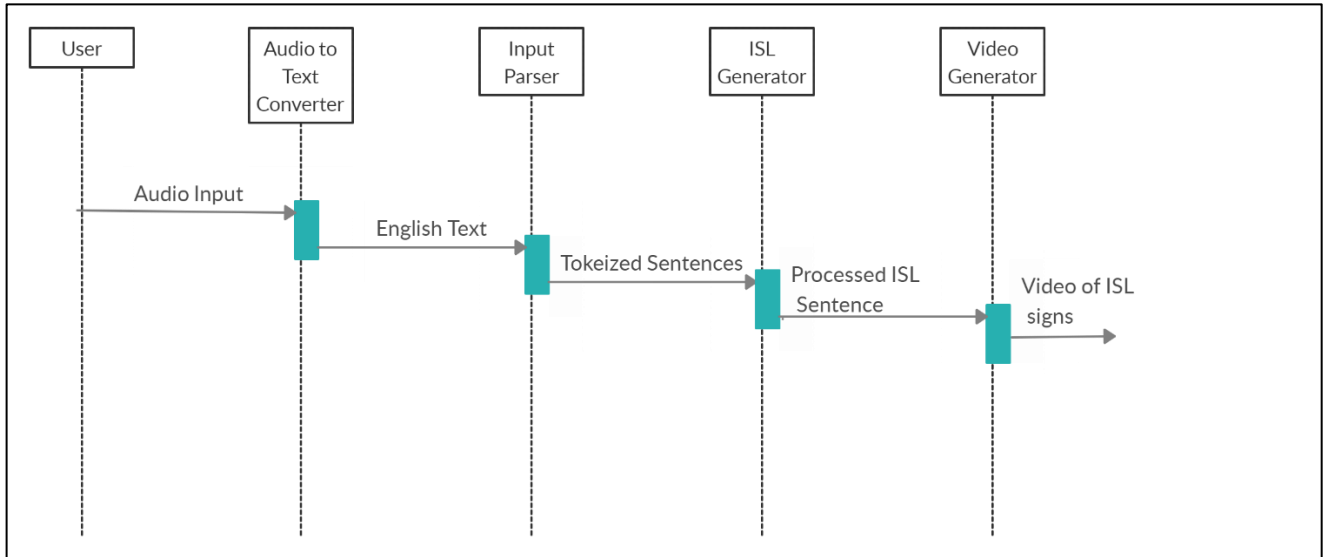


Figure 4.2 Sequence Diagram

4.3 GUI Design

This section consists of some screenshot of the project. The project is deployed as a website. The website has two sections, “Demo” section and “About” section. In Demo section, the user can give the audio input and the corresponding video with ISL signs and ISL sentence will be shown. In About section, there is information about the website. The following images show screenshots Demo sections and About section of website.

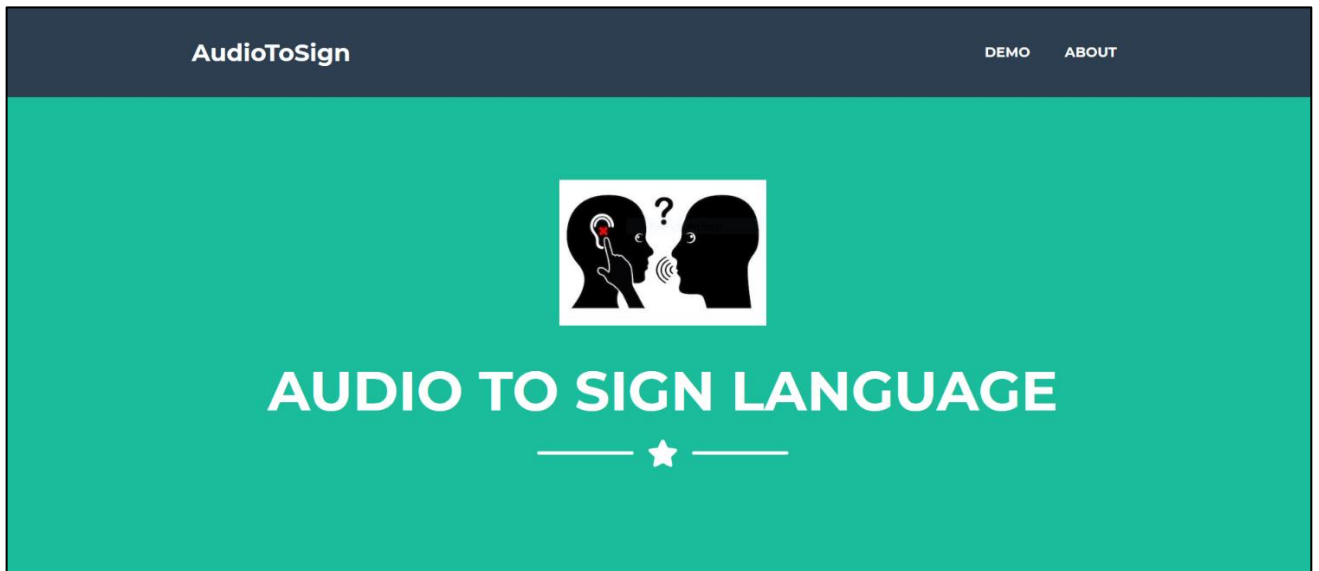


Figure 4.3 Landing Page

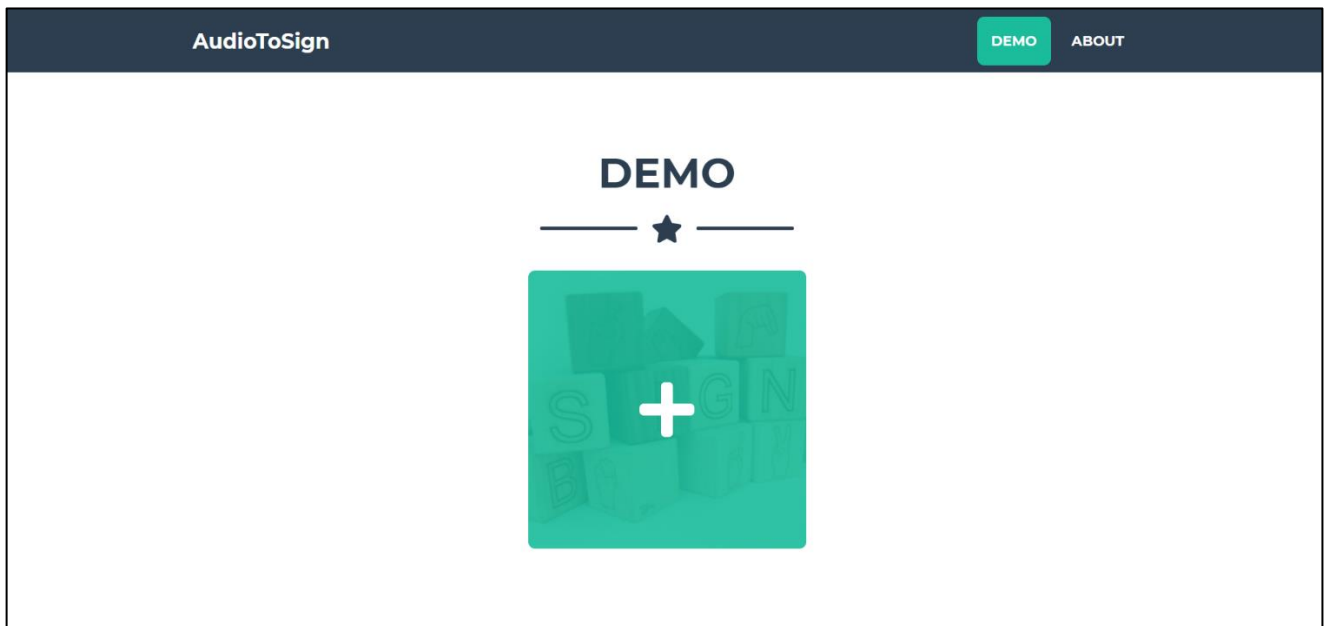


Figure 4.4 Demo Section Part 1

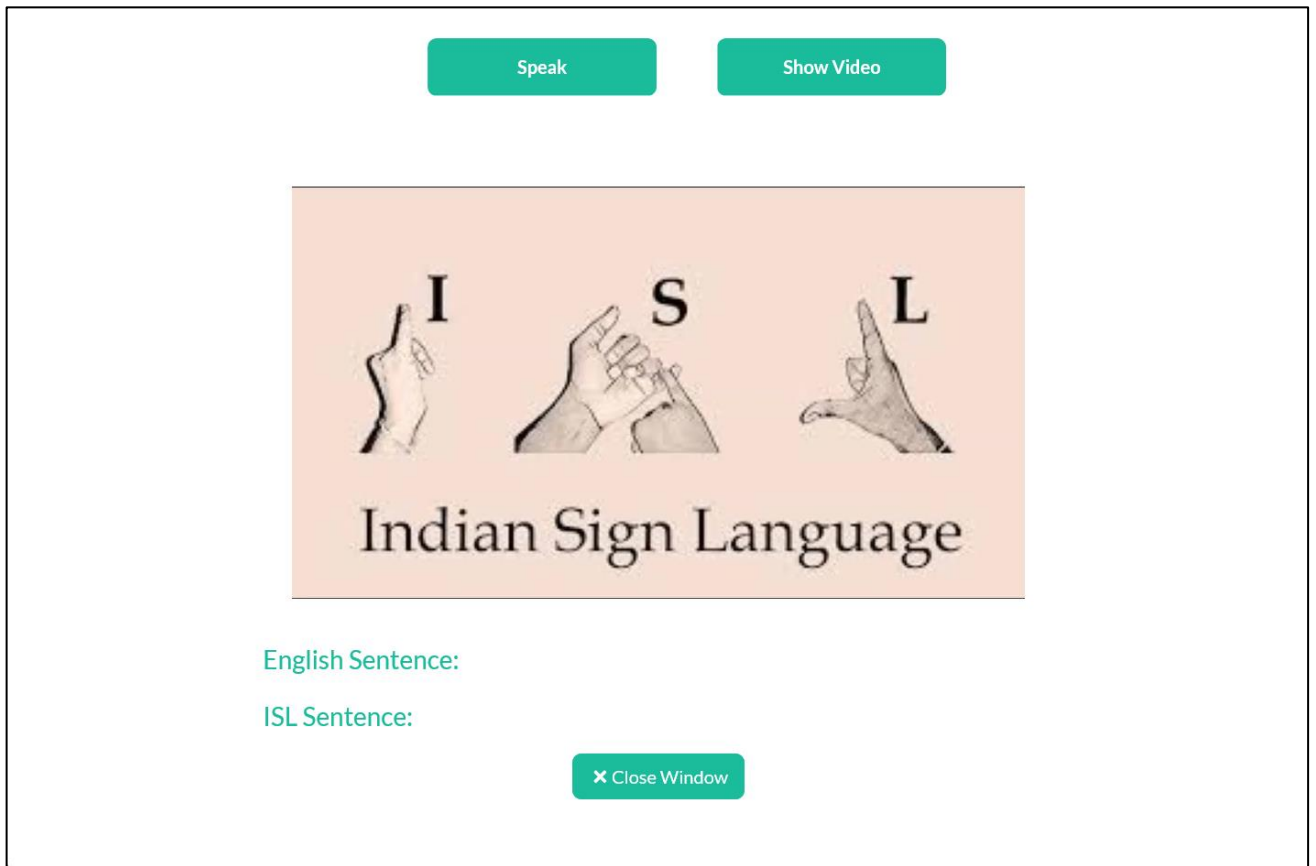


Figure 4.5 Demo Section Part 2

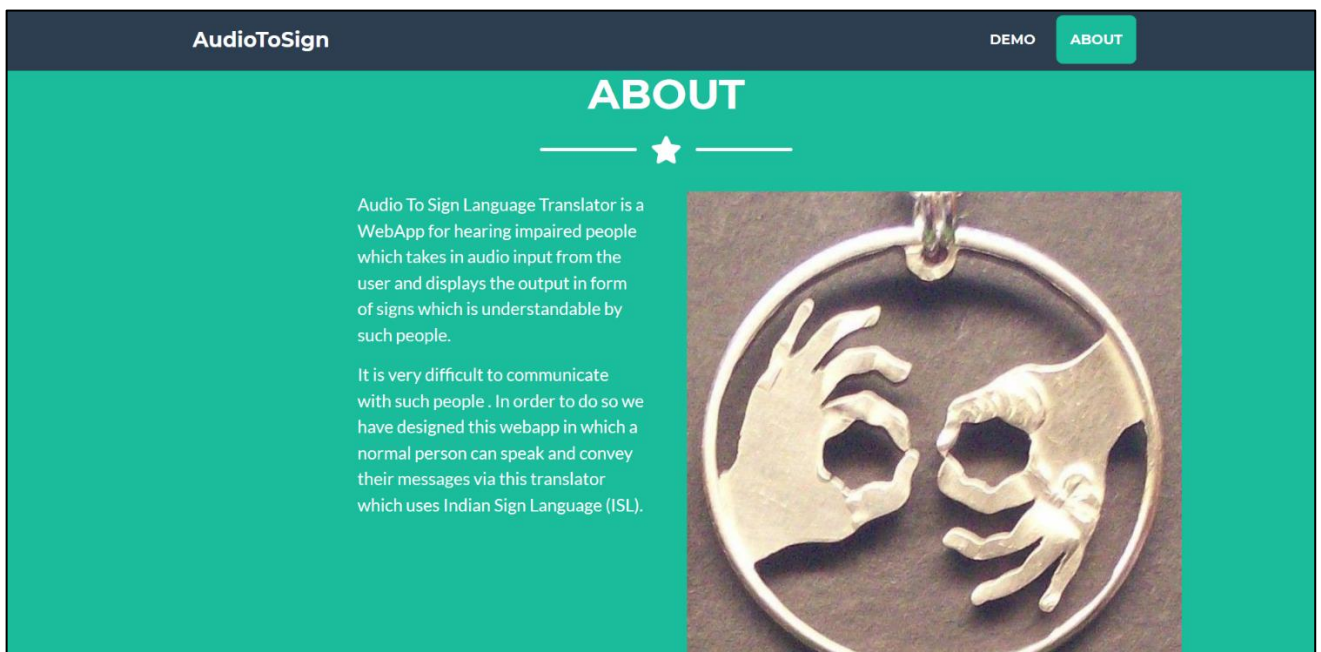


Figure 4.6 About Section

Chapter 5

Implementation and Results

This chapter contains detailed explanation about the implementation and result analysis of the project. In implementation details, every step of the implementation is explained with output. In result and evaluation section, output screenshots of an example are shown.

5.1 Implementation Details

The project is implemented as a website. There are two buttons “Speak” and “Show Video”. After clicking “Speak” button, the user has to speak English sentence through the microphone. For viewing the converted ISL sentence and video, the user has to click “Show Video” button. The website is built in Python using Flask framework. In this section, module wise implementation is discussed.

5.1.1 Audio to Text Converter

After clicking “Speak” button, the audio input is given by the user through Microphone. For capturing the audio and processing it, we have used two modules of python “PyAudio” and “SpeechRecognition”. The audio is captured, noise is removed from the audio and then Google Speech API is used for converting the audio to text. The below image shows the steps for audio to text conversion.

```
import speech_recognition as sr

r = sr.Recognizer()
with sr.Microphone() as source:                # recognize the microphone
    r.adjust_for_ambient_noise(source)          # noise removal
    print("Say:")
    audio = r.listen(source)                   # capturing audio
    input_string = r.recognize_google(audio)     # converted input text
```

Figure 5.1 Audio to Text Conversion

The “SpeechRecognition” module internally uses **Hidden Markov Model (HMM)** algorithm for converting audio to text. The physical sound given by the user is converted to electrical signal via the Microphone and then to digital signal by the analog to digital converter. This process is done internally. Once digitized data is obtained, the following steps are performed in HMM algorithm

1. Speech signal is divided into fragments of 10 milliseconds each.
2. The power spectrum of each fragment, which is essentially a plot of signal’s power as a function of frequency, is mapped to a vector of real numbers.
3. To decode the speech into text, groups of vectors are matched to one or more phonemes: a fundamental unit of speech.
4. A special algorithm is then applied to determine the most likely word (or words) that produce the given sequence of phonemes.
5. The final output of the HMM is a sequence of these vectors.

The English text sentence is then passed to the next module for parsing.

5.1.2 Input Parser

The input text is parsed and a phrase structure tree is formed. We have used two main modules of NLTK, “CoreNLPServer” and “CoreNLPParser”. In addition, we have also used “Tree” module of NLTK for creating the tree structure. The Natural Language Toolkit (NLTK) is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. Stanford CoreNLP provides a set of human language technology tools. It can give the base forms of words, their parts of speech, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, get the quotes people said, etc.

CoreNLPServer includes a simple web API server for servicing human language understanding needs. In our project we have created a CoreNLPServer instance. Before loading the website, one has to make sure that the server is running in the backend, only then the services of NLP can be used. CoreNLPParser provides full syntactic analysis, a constituency (phrase-structure tree) parse of sentences. In our project, we have created an instance of CoreNLPParser for creating the phrase structure tree. Tree module of NLTK phrase structure a tree like structure.

For running CoreNLPServer, we had to download the Stanford CoreNLP modules from the official website. One should have Java 1.8 installed in machine for running the server. The following images show an example code of creating CoreNLPServer and CoreNLPParser instances.

```
from nltk.parse.corenlp import CoreNLPServer
import os

java_path = "Java_exe_file_path"          # java.exe file path
os.environ['JAVAHOME'] = java_path         # environment variable named JAVAHOME
STANFORD = os.path.join("models", "stanford-corenlp-full-2018-10-05")
server = CoreNLPServer(
    os.path.join(STANFORD, "stanford-corenlp-3.9.2.jar"),
    os.path.join(STANFORD, "stanford-corenlp-3.9.2-models.jar"),
)
server.start()                             # running the server
```

Figure 5.2 CoreNLPServer Implementation

```

from nltk.parse.corenlp import CoreNLPParser
from nltk.tree import *

input_string = "I ate an apple"

# initializing the parser
parser = CoreNLPParser()

# Generates all possible parse trees sort by probability for the sentence
possible_parse_tree_list = [tree for tree in parser.parse(input_string.split())]

# Get most probable parse tree
probable_tree = possible_parse_tree_list[0]

# Convert into tree data structure
parse_tree = ParentedTree.convert(probable_tree)

```

Figure 5.3 CoreNLPParser Implementation

Input string: "I ate an apple"

Output parse tree: (ROOT (S (NP (PRP I)) (VP (VBD ate) (NP (DT an) (NN apple)))))

5.1.3 ISL Generator

The phrase structure tree received in the previous step is modified in this step according to the ISL grammar rules mentioned in Chapter 3 section 3.3.3. As per the rule, in the phrase structure tree the noun phrase and the prepositional phrase are freezed but if there any verb phrase present in the tree, then it is checked recursively because the verb phrase may further be composed of noun phrase, prepositional phrase, verb phrase or even the sentence. Hence we recursively check the verb phrases in the parse tree and modify them. After getting the modified tree, in which only the leaf nodes contain the actual words of the ISL sentence, all the leaf nodes i.e. words are stored in a list in left to right sequence. This is the list of tokens of the converted ISL sentence.

This list of tokens is sent for Lemmatization. For lemmatization, we have used "WordLemmatizer" module of NLTK. After the tokens are being lemmatized, the modified list of tokens is sent for elimination of stop words. The following images show an example code of elimination of stop words and lemmatization.

```

from nltk.stem import WordNetLemmatizer

isl_token_list = ['I', 'apple', 'ate']
lemmatizer = WordNetLemmatizer()
lemmatized_words = [] # new list of lemmatized words
for token in isl_token_list:
    lemmatized_words.append(lemmatizer.lemmatize(token))

```

Figure 5.4 Lemmatizing Tokens

```

isl_token_list = ['I', 'an', 'apple', 'ate'] # ISL token list
stopwords_set = set(['a', 'an', 'the', 'is']) # set of stop words
# using lambda function to remove stop words from token list
isl_token_list = list(filter(lambda x: x not in stopwords_set, isl_token_list))

```

Figure 5.5 Filtering Stop Words

After eliminating the stop words, the final ISL sentence is formed by concatenating the tokens in the ISL token list. Then the ISL token list is sent for processing where we check for each token whether there is a video of that token (word) available in our ISL video dataset. We have created a list of words in which the names of all the videos available in our ISL video dataset is listed. We check whether each token is present in the list. If the token is present then we keep that token as it is. But if the token is not present in the list of words, then we break the token (word) into letters. This way we check for each token and create the processed ISL string.

The reason behind doing processing is if the token is present in word list, then a video of the same is present in the dataset which will be shown directly. And if the token is not present, it indicates that video of that token is not available in our ISL video dataset and the word has to be shown letter by letter. For example, if token is “*Rohit*” and this word is not present in the word list, then in final video output, “*Rohit*” will be shown as a sequence of signs of the letters “*r o h i t*”. The following images show the python list of words and an example code of processing of ISL token list.


```
dataset_words = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
'about', 'above', 'across', 'actress', 'address', 'aeroplane',
'after', 'again', 'alive', 'allergy', 'aluminium', 'always',
'apple', 'area', 'around', 'assembly', 'baby', 'badminton',
'ball', 'bandh', 'basketball', 'bat', 'before', 'behind', 'bell',
'below', 'bengali', 'between', 'birth', 'bloodpressure', 'boat',
'bogie', 'boxing', 'boy', 'brass', 'brother', 'cancer', 'capital',
'carrom', 'cement', 'centimeter', 'chess', 'cholera', 'christmas',
'coal', 'coconut', 'cold', 'corn', 'cough', 'cremate', 'cure',
'diabetes', 'die', 'disease', 'down', 'durga', 'during',
'election', 'engine', 'exercise', 'far', 'flag', 'foot',
'friend', 'from', 'fruit', 'funeral', 'game', 'ghost', 'government',
'grain', 'gram', 'grandfather', 'grandmother', 'grandson',
'groundut', 'guard', 'gujarati', 'health', 'helicopter', 'here',
'hindi', 'history', 'hockey', 'holi', 'housefly', 'inch',
'jackfruit', 'jaundice', 'karate', 'kilogram', 'kilometer',
'king', 'kite', 'leader', 'leather', 'lemon', 'leprosy',
'lighthouse', 'litre', 'lose', 'magnet', 'malayalam', 'marathi',
'mary', 'measles', 'measure', 'measurement', 'meter', 'mile',
'mililitre', 'minister', 'mug', 'mums', 'mumps', 'near', 'neighbour',
'newspaper', 'olympics', 'out', 'over', 'parliament', 'plastic',
'poison', 'pomegranate', 'postman', 'power', 'pray', 'president',
'priest', 'queen', 'race', 'railway', 'relative', 'reservation',
'rice', 'riots', 'rocket', 'room', 'rubber', 'run', 'scale', 'seat',
'ship', 'sick', 'silk', 'silver', 'sometime', 'son', 'steel',
'strike', 'surname', 'tamil', 'telgu', 'temperature', 'there',
'through', 'ticket', 'toilet', 'towards', 'trophy', 'twin', 'under',
'urdu', 'vote', 'wedding', 'weight', 'wheat', 'wife', 'win', 'with',
'without', 'woman', 'wood', 'wool', 'wound']
```

Figure 5.6 Dataset Words

```
isl_sentence = "I apple ate" # converted ISL sentence
words = list(isl_sentence.split()) # splitting ISL sentence by whitespace
processed_isl_sentence = "" # processed ISL string

for word in words:
    if word not in dataset_words:
        for letter in word:
            processed_isl_sentence += " " + letter
    else:
        processed_isl_sentence += " " + word
```

Figure 5.7 Processing of ISL Sentence

The output of each step is show below in a sequence

1. Audio to text conversion (input string): “I ate an apple”
2. Parse tree of English string: (ROOT (S (NP (PRP I)) (VP (VBD ate) (NP (DT an) (NN apple))))))

3. Modified parse tree (as per ISL rules): (ROOT (NP (PRP I)) (NP (DT an) (NN apple)) (VBD ate))
4. ISL token list: ['I', 'an', 'apple', 'ate']
5. Lemmatized token list: ['I', 'an', 'apple', 'ate']
6. Filtering stop words: ['I', 'apple', 'ate']
7. ISL sentence: "I apple ate"
8. Preprocessed ISL sentence: "i apple a t e"

5.1.4 Video Generation

The processed ISL sentence received from the previous module is used for making the final output video. The video is made by merging all the videos corresponding to each word or letter in the processed ISL sentence. We have used "MoviePy" module of python for merging the video files in a sequence and create a single video file. The following image shows an example code of merging videos.

```
from moviepy.editor import *

processed_isl_sentence = "i apple a t e"      # Processed ISL sentence
clips = []                                   # list which will hold the video files to be merged
l = processed_isl_sentence.split()           # splitting the string by whitespaces

#for every letter or word in processed ISL sentence,
#respective video file is added to clips[]
for i in l:
    clips.append(VideoFileClip(i+".mp4"))
result = concatenate_videoclips(clips)       # merging all video files in clips[]
result.write_videofile("result.mp4")         # creating video file and saving is as result.mp4
```

Figure 5.8 Merging Video Files

The merged video is displayed on the screen after clicking "Show Video" button. The spoken English sentence and the converted ISL sentence is shown on the screen after clicking the button.

5.2 Result and Evaluation

In this section, we have taken the same example as discussed in the previous section. First run the Core NLP Server, then the website is loaded in browser. After clicking the “Speak” button, audio input is given as “I ate an apple”. In the console, output of each step is showed i.e.

1. The converted audio text
2. The parse tree of the English sentence
3. The modified parse tree as per ISL rules
4. The ISL token list (leaf nodes of the modified parse tree)
5. Lemmatized token list
6. Token list after filtering stop words
7. The ISL sentence
8. The processed ISL sentence

After these outputs are shown in the console, the video output is created by merging videos of letters and words in processed ISL sentence. It will show a progress in percentage of creating the video. After the video is created, it gives a message “Done” in the console. After getting the “Done” message in the console, “Show Video” button is clicked which results in displaying the video of ISL signs and also the English sentence and ISL sentence.

```
Say:
I ate an apple

Parse Tree:

(ROOT (S (NP (PRP I)) (VP (VBD ate) (NP (DT an) (NN apple)))))

Modified Parse Tree:

(ROOT (NP (PRP I)) (NP (DT an) (NN apple)) (VBD ate))

ISL Token List:
['I', 'an', 'apple', 'ate']

Lemmatized Token List:
['I', 'an', 'apple', 'ate']

Filtered Stopwords:
['I', 'apple', 'ate']

ISL Sentence:
I apple ate

Processed ISL Sentence:
i apple a t e

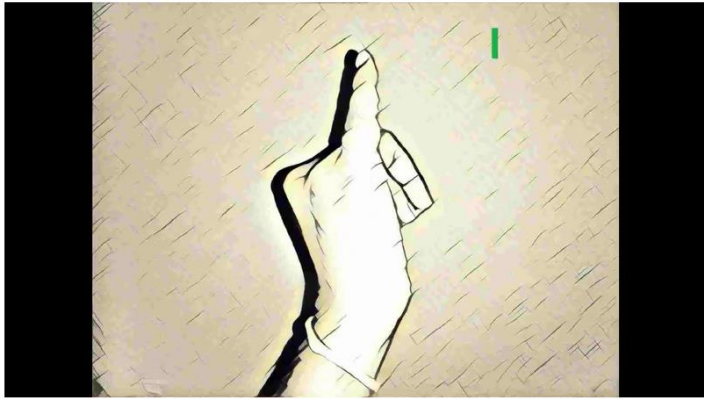
Moviepy - Building video static/result.mp4.
MoviePy - Writing audio in resultTEMP_MPY_wvf_snd.mp3
MoviePy - Done.
Moviepy - Writing video static/result.mp4

Moviepy - Done !
Moviepy - video ready static/result.mp4
127.0.0.1 - - [26/Apr/2020 20:34:24] "GET /speechConverter HTTP/1.1" 200 -
```

Figure 5.9 Console Output

Speak

Show Video



English Sentence:

I ate an apple

ISL Sentence:


i apple ate

✕ Close Window

(a)

Speak

Show Video



English Sentence:

I ate an apple

ISL Sentence:


i apple ate

✕ Close Window

(b)

Speak

Show Video



English Sentence:

I ate an apple

ISL Sentence:

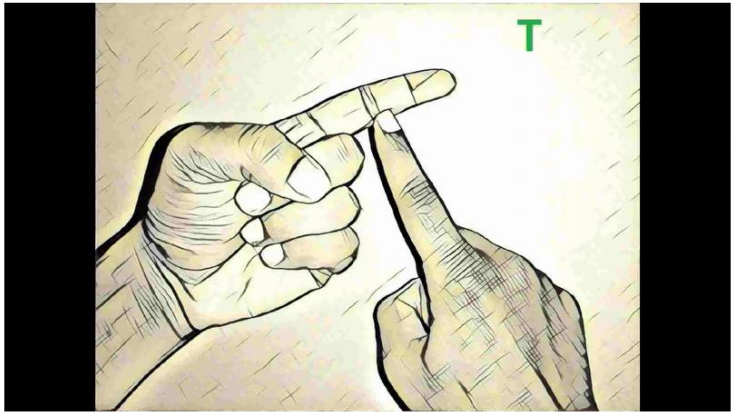
i apple ate

✕ Close Window

(c)

Speak

Show Video



English Sentence:

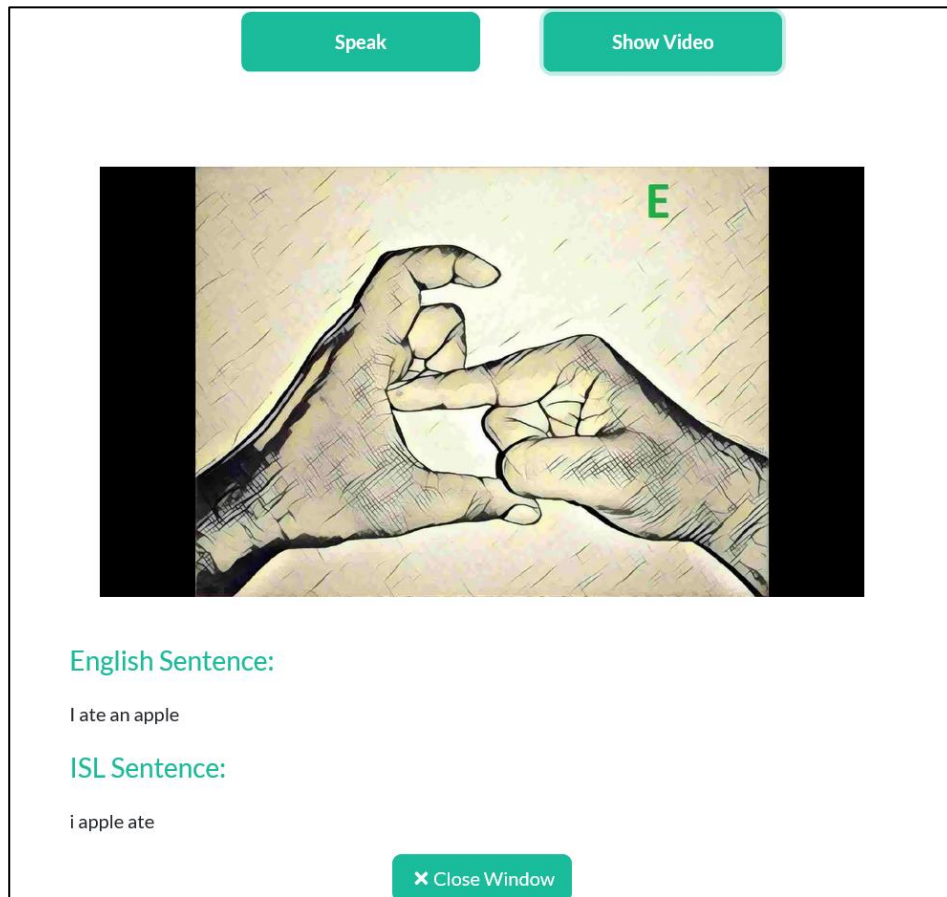
I ate an apple

ISL Sentence:

i apple ate

✕ Close Window

(d)



(e)

Figure 5.10 Screenshots of Output

The following points should be kept in mind while running the project.

1. The Core NLP Server should always run in the backend, else the English to ISL conversion will not take place properly.
2. The working of Core NLP Server will depend on the internet speed, hence sometimes there may be a delay in the output Core NLP Parser. Hence the user should have a good internet connection to get accurate result.
3. Once the “Speak” button is pressed, the user should not immediately start speaking to give the input until the Microphone icon is shown on the task bar or the word “Say” is shown on the command prompt. Every time the user clicks the button to give an input the user has to make sure of the above point.

4. The accuracy of the output signs will be of desired level only when the input audio is clear till an optimum level. Prediction/auto-correction of input words may differ the actual desired output.
5. In order to see the generated video of the input sentence, the user will have to click the “Show Video” button only when the video is 100% successfully generated on the command prompt.
6. If there is no video of the words of ISL sentence available in the ISL video dataset i.e. if the processed ISL sentence consists of only letters, then it might take little bit longer to merge and generate the output video.

Chapter 6

Conclusion and Future Work

This chapter consists of the conclusion and the future work of the project. The conclusion section gives a summary and the conclusion of the project. The future work section explains about how the project can be further extended.

6.1 Conclusion

The Audio to Sign Language Translator system is implemented as a website which is developed in Python using Flask framework. The user gives the audio input to the system which is successfully converted to English text using HMM algorithm internally. The English text is then converted to ISL sentence according to ISL grammar rules. This is done by performing various processes which are creating parse tree of English sentence, modifying the parse tree

as per ISL grammar rules, crating the token list, lemmatizing the tokens and filtering stop words. The ISL sentence is then processed for creating processed ISL text which is further used to generate the video of ISL signs. The video as well as the ISL sentence is shown to the user. This project has tried to reduce the gap between the normal speaking people and the hard hearing and hard speaking prople.

6.2 Future Work

The Audio to Sign Translator system currently converts English Language to Indian Sign Language. But it might be possible that the User of the system is comfortable speaking any other regional language such as Hindi, Spanish, French, etc. Also it might be possible that the Impaired User knows a sign language other that Indian Sign Language such as American Sign Language, British Sign Language, etc. This project can be further extended for different types of source and target languages. For example, English to American Sign Language Translation, English to British Sign Language Translation, French to American Sign Language Translation, etc. These types of translations can be different services which this website can provide to the hard hearing and hard speaking people all around the world.

To create other type of translation, one needs to add the videos of signs in target language to the video dataset. The grammar rules of the source language as well as the target need to be studied very thoroughly for creating the algorithm for translation. Specifically the change will be in the processes of creating the parse tree of the source language and modifying it to parse tree of target language as per grammar rules of target language. The lemmatization and filtration of stop words would remain the same. The Stanford Core NLP modules of the source language have to be downloaded if the source language is other than English. Hence, there is no need to create a separate website for another type of translation, different types of translations can be implemented as services of the same website.

References

- [1] M. Suresh Anand, M. Kumaresan, Dr. M. Mohan Kumar, “An Integrated Two Way ISL (Indian Sign Language) Translation System – A New Approach”, vol 4, No. 2, Jan – Feb 2013.
- [2] Becky Sue Parton, “Sign Language Recognition and Translation: A Multidisciplined Approach From the Field of Artificial Intelligence”, University of North Texas.
- [3] Rajaganapathy. S, Aravind. B, Keerthana. B, Shivgami. M, “Conversion of Sign Language to Speech with Human Gestures”, VIT University, Chennai.
- [4] Abey Abraham, Rohini V, “Real Time Conversion of Sign Language to Speech and Prediction of Gestures Using Artificial Neural Network”.
- [5] <https://www.talkinghands.co.in/>, visited on 20th April 2020.
- [6] <https://www.districtdatalabs.com/syntax-parsing-with-corenlp-and-nltk>, visited on 3rd April 2020.
- [7] <https://stanfordnlp.github.io/CoreNLP/parse.html>, visited on 26th March 2020.
- [8] <https://www.geeksforgeeks.org/project-idea-audio-sign-language-translator/>, visited on 3rd February 2020.

Acknowledgement

We would like to express our gratitude and thanks to **Prof. Jayant Gadge** for his valuable guidance and help. We are indebted for his guidance and constant supervision as well as for providing necessary information regarding the project. We would like to express our greatest appreciation to our principal **Dr. G. T. Thampi** and head of the department **Dr. Tanuja Sarode** for their encouragement and tremendous support. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of the project.

Juhi Gianani

Priya Gianchandani

Ankita Kar