

SPEECH TO SIGN LANGUAGE CONVERTOR

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in

Computer Science and

Engineering By

MOKESH.P (Reg. No 37110451)

MUKESHKUMAR.V (Reg. No 37110459)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI

SALAI, CHENNAI – 600 119

APRIL - 2021



**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY**

(Established under Section 3 of UGC Act, 1956)
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai -
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project is the bonafide work of **MOKESH. P (Reg.no: 37110451)** and **MUKESHKUMAR. V (Reg. no: 37110459)** who carried out the project entitled **“SPEECH TO SIGN LANGUAGE CONVERTOR”** under my supervision from **November 2021** to **April 2021**.

Internal Guide

Dr. P. Ajitha, M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

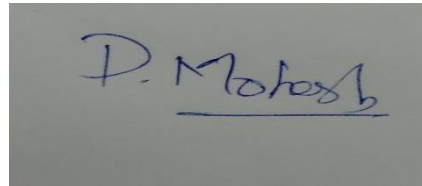
Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I, **MOKESH. P (Reg. No. 37110451)** hereby declare that the project report entitled on **“SPEECH TO SIGN LANGUAGE CONVERTOR”** done by me and my partner under the guidance of **Dr. P. Ajitha, M.E, Ph.D.**, Department of Computer Science and Engineering at Sathyabama has submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

A rectangular box containing a handwritten signature in blue ink. The signature appears to be 'P. Mokesh' with a horizontal line underneath the name.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge the sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneshwari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express the sincere and deep sense of gratitude to my Project Guide **Dr. P. Ajitha, M.E, Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of the project work.

I wish to express the thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

People find it difficult to get a source to learn various sign languages such as ISL (Indian Sign Language), ASL (American Sign Language) and many such languages. It is also a tedious work to find the proper learning material for all the words in sign language on internet. Some physically challenged people might also find it difficult to learn the sign language at their initial stage of learning. Sources available over internet is not sorted and easy to learn.

So to overcome this issue we came-up with a idea of creating a web application which will convert the input speech to corresponding Indian Sign Language in an effort to bridge the gap in the process of communication between the Dumb people and also would help people to learn ISL in an easier way through a website.

As a result this web application will give a sustainable solution to the end users of the website. So this project may be useful to educational institutions to make the students learn especially for the special students.

TABLE OF CONTENTS

ABSTRACT	v
LIST OF FIGURES	viii

CHAPTER NO.	TITLE	PAGE NO
1.	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 RESEARCH AND SIGNIFICANCE	1
2.	AIM AND SCOPE	2
	2.1 AIM OF THE PROJECT	2
	2.1.1 OBJECTIVES	2
	2.2 SCOPE	2
	2.2.1 ADVANTAGES	2
3.	SYSTEM DESIGN AND METHODOLOGY	3
	3.1 EXISTING SYSTEM	3
	3.2 SOFTWARE REQUIREMENTS	4
	3.3 HARDWARE REQUIREMENTS	4
	3.4 SYSTEM FUNCTIONALITIES	4
	3.5 TRAINING	5
	3.6 APPLICATION OF SIGN LANGUAGE	6
4.	APPLICATION DEVELOPMENT METHODOLOGY	8
	4.1 LANGUAGE USED FOR WEB APPLICATION	8
	4.2 THE STAGES OF SOFTWARE DEVELOPMENT LIFECYCLE	10
	4.2.1 PLANNING AND REQUIREMENT	10

	ANALYSIS	
	4.2.2 DEFINING REQUIREMENTS	11
	4.2.3 DESIGNING THE PRODUCT ARCHITECTURE	11
	4.2.4 BUILDING OR DEVELOPING THE PRODUCT	11
	4.2.5 TESTING THE PRODUCT	11
	4.2.6 DEPLOYMENT IN THE MARKET AND MAINTENANCE	12
5.	RESULTS AND DISCUSSION	13
	5.1 RESULTS	14
6.	CONCLUSION AND FUTURE WORK	17
	6.1 CONCLUSION	17
	6.2 FUTURE WORK	18
	REFERENCES	19
	APPENDIX	21
	A. SOURCE CODE	21

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
4.1	WORKING DIAGRAM	12
5.1	TEXT INPUT	13
5.2	CORRESPONDING SIGN LANGUAGE	14
5.3	SPEECH INPUT	15
5.4	CORRESPONDING SIGN LANGUAGE OUTPUT	16
6.1	FLOW OF THE PROJECT	17

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

My project idea is to create an WEB application that can convert the speech input into the corresponding sign language. Some parents of normal hearing children teach their infants signed English. Advocates believe babies can learn to communicate their needs – such as being hungry or thirsty – through the use of signs before they are able to speak. Scientists believe children who learn a second language when they are very young develop better language skills. Due to its visual nature, sign language is a great tool for early readers and enhances spelling skills. When someone you love can't hear, ISL is a great way to communicate in a rich, meaningful way. It's also the best way to develop awareness and sensitivity to the Deaf culture, a community of non-hearing individuals which number more than one million in the United States alone. Whether you teach your baby to sign or learn ISL to communicate with a deaf friend or family member, you are using a full-bodied form of communication that will enhance your relationship as it improves your mind and spirit.

1.2 RESEARCH AND SIGNIFICANCE

The main goal of this project is to convert the speech or text into the corresponding Indian Sign Language. The project will consist of several components such as

- PYTHON
- JAVASCRIPT
- HTML
- CSS

The main idea used here is create a web application that can convert the text or the speech to sign language. This is achieved using the above mentioned python,JS,CSS,HTML.

CHAPTER 2

AIM AND SCOPE

2.1 AIM OF PROJECT

The primary aim of the proposed system is to convert the speech input or the text input into the corresponding ISL (Indian Sign Language) convertor. The input can be an alphabet or an integer or a sentence.

2.1.1 OBJECTIVES

The proposed model “Speech to Sign Language Convertor” is based on the web application that can convert any speech or text input into the corresponding Indian Sign Language. To create an application that helps people to change the voice input into corresponding sign language by changing the voice input into text and then the respective text to sign language. To achieve these objective certain obstacles were faced by us such as getting datasets, training models with maximum accuracy. So, to overcome this obstacle we created our own datasets and fed them and we believe our end results would be efficient enough.

2.2 SCOPE

This model will help in converting the speech to the ISL (Indian Sign Language) This will be helpful for the people who wants to learn ISL and also will help some physically challenged people might also find it difficult to learn the sign language at their initial stage of learning. Sources available over internet is not sorted and easy to learn.

2.2.1 ADVANTAGES

- Used to learn the ISL language easily.
- It can help physically challenged people to communicate with each other.
- It is used to provide the speech to the corresponding sign language

CHAPTER 3

SYSTEM DESIGN & METHODOLOGY

3.1 EXISTING SYSTEM

The objective of this work is to annotate sign instances across a broad vocabulary in continuous sign language. We train a Transformer model to ingest a continuous signing stream and output a sequence of written tokens on a large scale collection of signing footage with weakly-aligned subtitles. We show that through this training it acquires the ability to attend to a large vocabulary of sign instances in the input sequence, enabling their localisation. Our contributions are as follows we demonstrate the ability to leverage large quantities of continuous signing videos with weakly-aligned subtitles to localise signs in continuous sign language we employ the learned attention to automatically generate hundreds of thousands of annotations for a large sign vocabulary we collect a set of 37K manually verified sign instances across a vocabulary of 950 sign classes to support our study of sign language recognition by training on the newly annotated data from our method, we outperform the prior state of the art on the sign language recognition benchmark.

Sign Language is the dominant yet non-primary form of communication language used in the deaf and hearing impaired community. To make an easy and mutual communication between the hearing-impaired and the hearing communities, building a robust system capable of translating the spoken language into sign language and vice versa is fundamental. To this end, sign language recognition and production are two necessary parts for making such a two-way system. Sign language recognition and production need to cope with some critical challenges. In this survey, we review recent advances in Sign Language Production (SLP) and related areas using deep learning. This survey aims to briefly summarize recent achievements in SLP,

discussing their advantages, limitations, and future directions of research.

3.2 SOFTWARE REQUIREMENTS

The basic software requirements to run the program are:

- 1) Windows 10
- 2) Any Browser eg. Mozilla Firefox, Chrome, Safari, etc.
- 3) Python
- 4) HTML
- 5) CSS
- 6) JavaScript
- 7) Speech Recognition Package

3.3 HARDWARE REQUIREMENTS

The basic hardware required to run the program are:

- 1) Minimum system memory
- 2) Computer with internet connection
- 3) Microphone

3.4 SYSTEM FUNCTIONALITIES

- 1) **PYTHON** : Used for both front end and backend
- 2) **HTML, CSS, JAVASCRIPT** : Used for creating a frontend.
- 3) **BROWSER** : Used for opening the web application.
- 4) **RECORDING** : Microphone is used for recording the speech input.

New deep-learning architectures are created every year, achieving state-of-the-art results in image recognition and leading to the belief that, in a few years, complex tasks such as sign language translation will be considerably easier, serving as a communication tool for the hearing-impaired community. On the other hand, these algorithms still need a lot of data to be trained and the dataset creation process is

expensive, time-consuming, and slow. Thereby, this work aims to investigate techniques of digital image processing and machine learning that can be used to

create a sign language dataset effectively. We argue about data acquisition, such as the frames per second rate to capture or subsample the videos, the background type, pre processing, and data augmentation, using convolutional neural networks and object detection to create an image classifier and comparing the results based on statistical tests. Different datasets were created to test the hypotheses, containing 14 words used daily and recorded by different smartphones in the RGB colour system. We achieved an accuracy of 96.38% on the test set and 81.36% on the validation set containing more challenging conditions, showing that 30 FPS is the best frame rate subsample to train the classifier, geometric transformations work better than intensity transformations, and artificial background creation is not effective to model generalization. These trade-offs should be considered in future work as a cost-benefit guideline between computational cost and accuracy gain when creating a dataset and training a sign recognition model.

3.5 TRAINING

We show that geometric transformations are more effective than intensity transformations, since the model infers the sign based on the hand shape and position in the image, also being invariant to the background type. We also argued that 30 FPS was the best rate to record the videos of the dataset, but it depends on the context such as the speed of the sign execution and lighting conditions. Equally, when training the model 30 FPS was a better subsample choice but it should vary depending on how the spatial- temporal features are explored. Finally, we created a multi-stream CNN architecture, based on our findings, that achieved an accuracy of 96% on the test set and 81% on the validation set. In future works, a larger dataset can be created, considering the findings of this paper and using different techniques for classification such as a spatial- temporal classifier and ensemble methods.

Independent Sign Language Recognition is a complex visual recognition problem that combines several challenging tasks of Computer Vision due to the necessity to

exploit and fuse information from hand gestures, body features and facial expressions. While many state-of-the-art works have managed to deeply elaborate on these features

independently, to the best of our knowledge, no work has adequately combined all three information channels to efficiently recognize Sign Language. In this work, we employ SMPL-X, a contemporary parametric model that enables joint extraction of 3D body shape, face and hands information from a single image. We use this holistic 3D reconstruction for SLR, demonstrating that it leads to higher accuracy than recognition from raw RGB images and their optical flow fed into the state-of-the-art l3D-type network for 3D action recognition and from 2D Open pose skeletons fed into a Recurrent Neural Network. Finally, a set of experiments on the body, face and hand features showed that neglecting any of these, significantly reduces the classification accuracy, proving the importance of jointly modelling body shape, facial expression and hand pose for Sign Language Recognition. we investigated the extraction of 3D body pose, face and hand features for the task of Sign Language Recognition.

Sign language is a gesture based symbolic communication medium among speech and hearing impaired people. It also serves as a communication bridge between non-impaired population and impaired population. Unfortunately, in most situations a non-impaired person is not well conversant in such symbolic languages which restricts natural information flow between these two categories of population. Therefore, an automated translation mechanism can be greatly useful that can seamlessly translate sign language into natural language.

3.6 APPLICATION OF SIGN LANGUAGE

Sign language recognition (SLR) is a challenging, but highly important research field for several computer vision systems that attempt to facilitate the communication among the deaf and hearing impaired people. In this work, we propose an accurate and robust deep learning-based methodology for sign language recognition from video sequences. Our novel method relies on hand and body skeletal features extracted from RGB videos and, therefore, it acquires highly discriminative for

gesture recognition skeletal data without the need for any additional equipment, such as data gloves, that may restrict signer's movements. Experimentation on a large publicly available sign language dataset reveals the superiority of our methodology with

respect to other state of the art approaches relying solely on RGB features.

Sign language is a kind of important communication gesture to be studied in the human-computer interaction field. Kinect is a 3D somatosensory camera launched by Microsoft, which can capture the colour, depth and skeleton frames and is helpful to the gesture recognition research. In this paper, a method using the HOG and SVM algorithms with the Kinect software libraries to recognize sign language by recognizing the hand position, hand shape and hand action features is proposed. In order to realize this method, a special 3D sign language dataset contains 72 words is collected with Kinect, and experiments are conducted to evaluate the method. It is shown in the experimental results that the use of the HOG and SVM algorithms significantly increases the recognition accuracy of the Kinect, and is insensitive to background and other factors. The average recognition rate is up to 89.8%, which means the Kinect-based recognition method proposed in this paper can effectively and efficiently recognize sign language, and it has a great significance to the research and promotion of the sign language recognition technology.

The Sign language is a visual language used by the people with the speech and hearing disabilities for communication in their daily conversation activities. It is completely an optical communication language through its native grammar, be unlike fundamentally from that of oral languages. In this research paper, presented an optimal approach, whose major objective is to accomplish the transliteration of 24 static sign language alphabets and numbers of American Sign Language into humanoid or machine decipherable English manuscript. Pre-processing operations of the signed input gesture are done in the first phase. In the next phase, the various region properties of pre-processed gesture image are computed. In the final phase, based on the properties calculated of earlier phase, the transliteration of signed gesture into text has been carried out. This paper also presents the statistical result

evaluation with the comparative graphical depiction of existing techniques and proposed technique.

CHAPTER 4

APPLICATION DEVELOPMENT METHODOLOGY

4.1 LANGUAGE USED FOR WEB APPLICATION

The language used for this project is python. This is an open source product. To know more about the software we can look into the basics of the language.

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system with reference counting.

Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.5. and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, a free and open-source reference

implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Commonly used algorithms compute the statistical probability that each pixel belongs to each class. Pixels are then assigned to the class associated with the highest probability. Algorithms of this kind are known as maximum likelihood. The result is an image in which every pixel has been assigned to one of two land cover classes.

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6. and 2.7.x version series. Releases of Python 3 include

the  utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.

Speech Recogniser : Speech Recogniser is used to recognise the speech input and do the corresponding process such as conversion to the text, detection of speech. This helps people to recognise the speech using this package that can be imported through python. This makes technological devices more accessible and easier to use.

PyAudio : PyAudio is a set of Python bindings for Port Audio, a cross-platform C++ library interfacing with audio drivers.

PyAudio: is used to access the microphone and detect the input spoken in the form of speech.

HTML: HTML can be expanded as Hyper Text Markup Language. It is used for web designing. Here HTML is used to create the front end of the website.

FLASK: Flask is a micro web framework written in Python. Flask is used here for hosting server in backend.

JAVASCRIPT: JavaScript is a scripting or programming language that allows you to implement complex features on web pages. Here JavaScript is used to store the voice in the webpage.

4.2 THE STAGES OF SOFTWARE DEVELOPMENT LIFECYCLE (SDLC)

4.2.1 Planning and Requirement Analysis: Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas. Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning

stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

4.2.2 Defining Requirements: Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

4.2.3 Designing the product architecture: SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification. This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity , budget and time constraints , the best design approach is selected for the product. A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

4.2.4 Building or Developing the Product: In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle. Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

4.2.5 Testing the Product: This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages

of SDLC. However, this stage refers to the testing only stage of the product where products defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

4.3.6 Deployment in the Market and Maintenance: Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations. business strategy. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

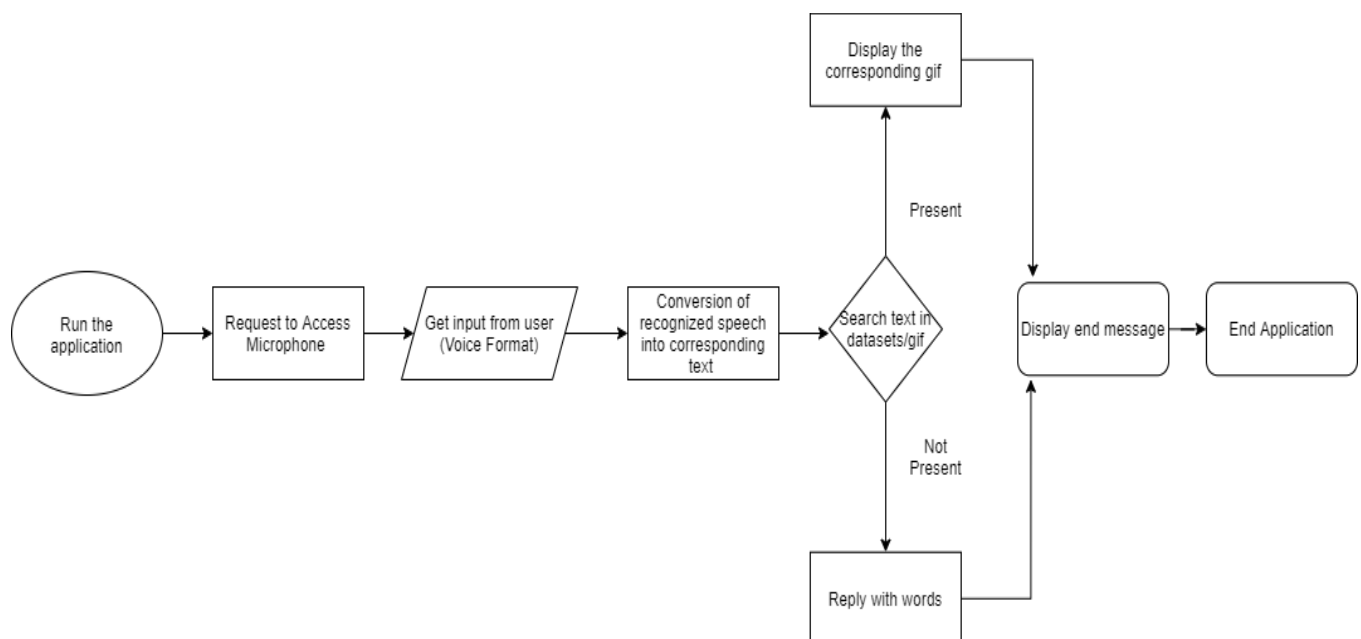


Fig 4.1: working diagram

CHAPTER 5

RESULTS AND DISCUSSION

5.1 RESULTS

Initially the user inputs the data in the form of voice and then this voice is converted into text using speech recognition technology. Then the converted text is fed for search into the datasets available. If the text matches the corresponding set of gif, then they are displayed. If they does not match then they are displayed in the format of single alphabets. The system becomes more user friendly when it is Speech to Sign Language Translation System. This can be used to teach small hearing- impaired kids who cannot read text. This can very well ease the conversation gap, creating more job opportunities and education opportunities for the hearing impaired as the requirement of the knowledge of sign language is no longer a criterion to teach hearing impaired after this software develops further.

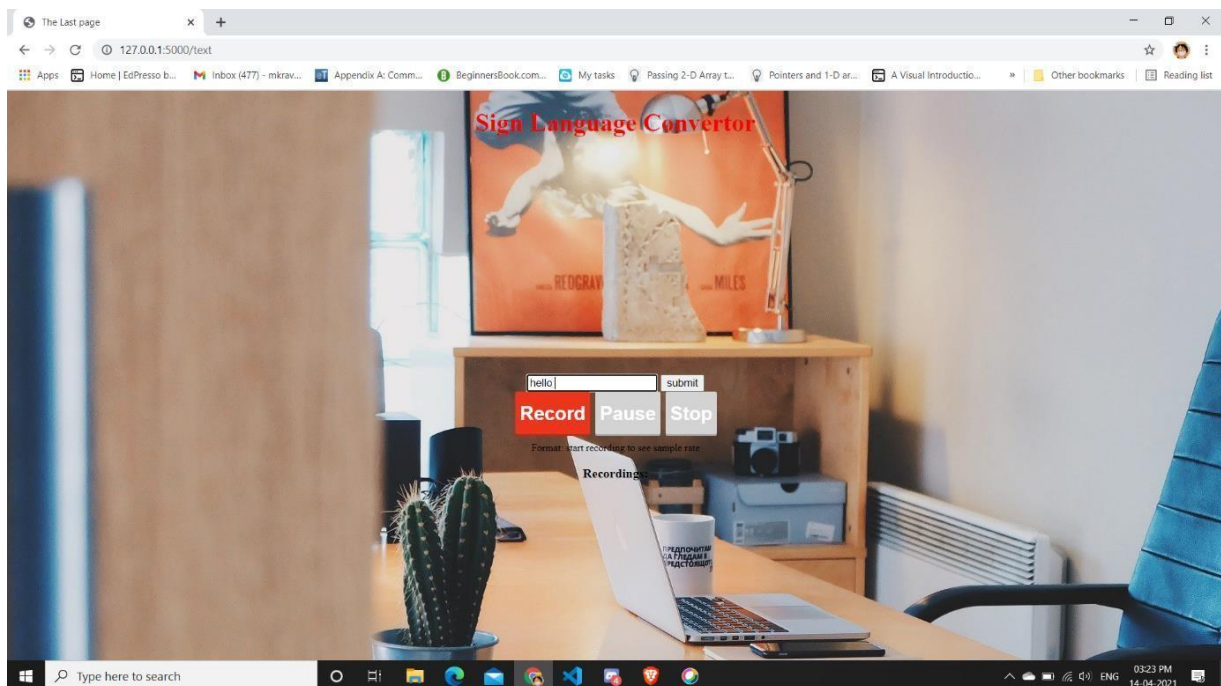


Fig 5.1: Text Input

The system requires the following to take input in form of dialogue python.

Flask is the web framework deployed here. HTML enhances better interface between the model and the user. It will come handy for visually challenged people to know the scenario and also for the people who like to learn ISL.

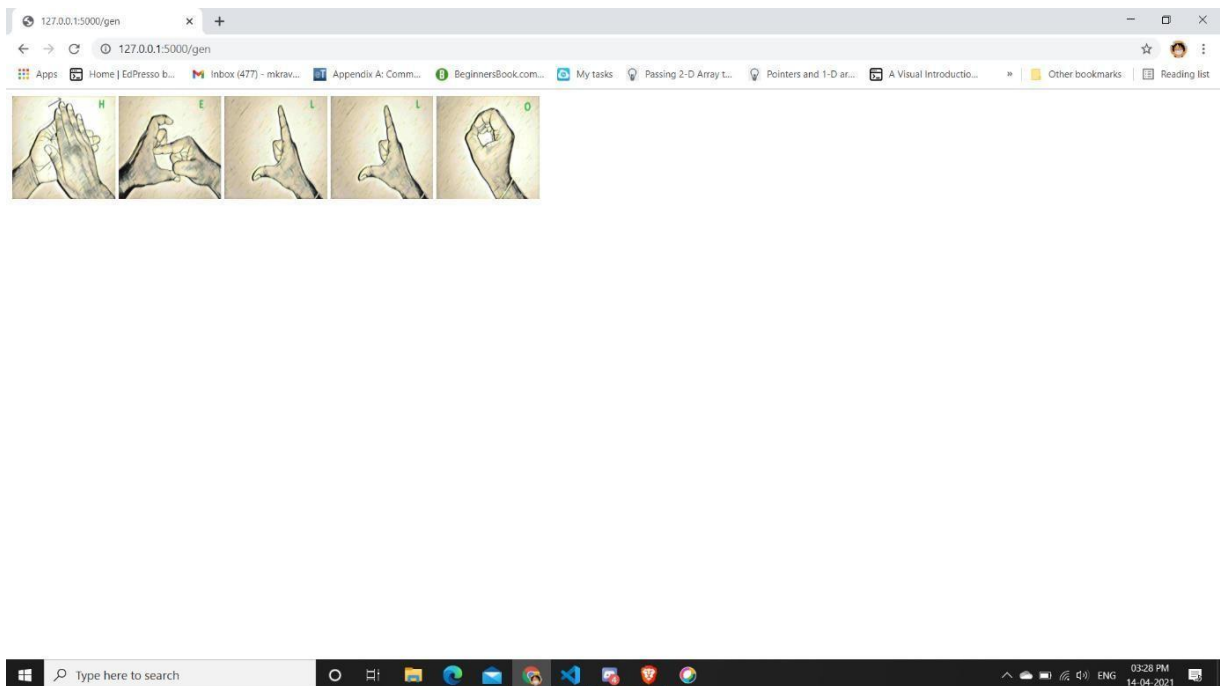


Fig 5.2: Corresponding Sign Language

The outcome of this project is a website that produces sign language based on the input speech provided. Initially the user inputs the data in the form of voice and then this voice is converted into text using speech recognition technology.

Then the converted text is fed for search into the datasets available. If the text matches the corresponding set of data, then they are displayed. If they do not match then they are displayed in the format of single alphabets.

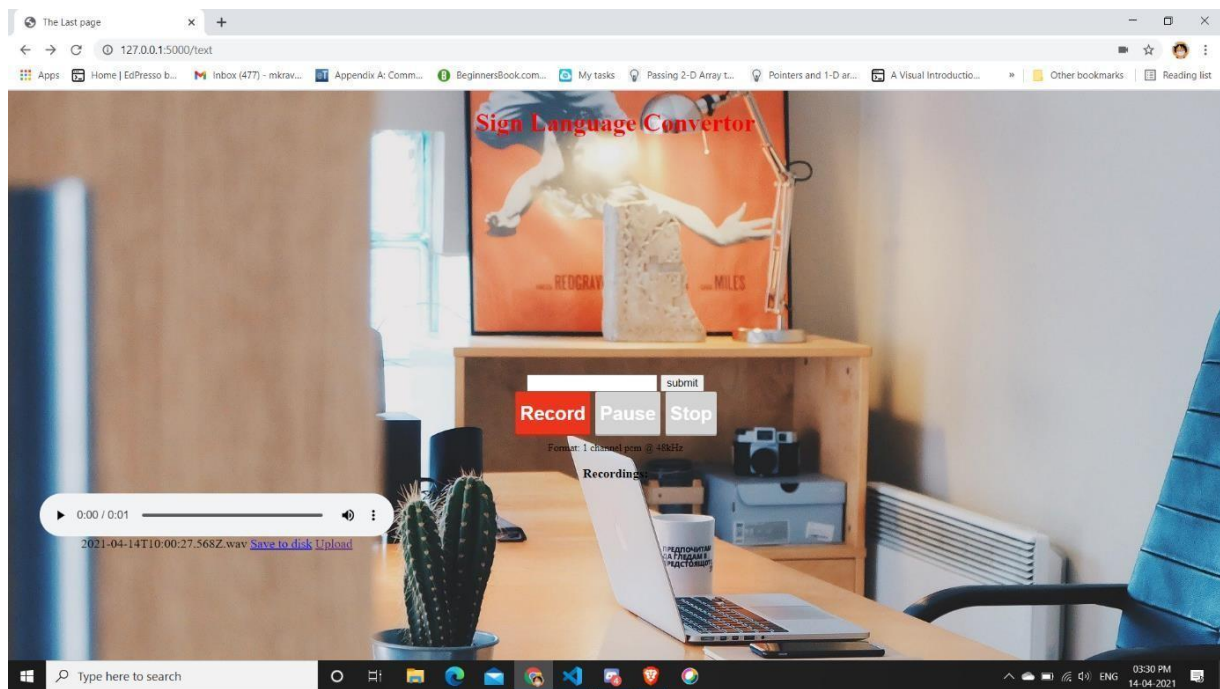


Fig 5.3: Speech Input

Here the input is in format of speech which is recorded by the application then it is converted into the corresponding sign language.

We used Speech Recognition to recognize the input voice to the corresponding text. Then using datasets available we convert the corresponding text into sign language. Then we used HTML, JavaScript to create web application and made our project API to work on that particular web application .

Initially the user inputs the data in the form of voice and then this voice is converted into text using speech recognition technology. Then the converted text is fed for search into the datasets available. If the text matches the corresponding set of gif, then they are displayed. If they does not match then they are displayed in the format of single alphabets. The specialty about this project is that this helps to learn Indian Sign Language.

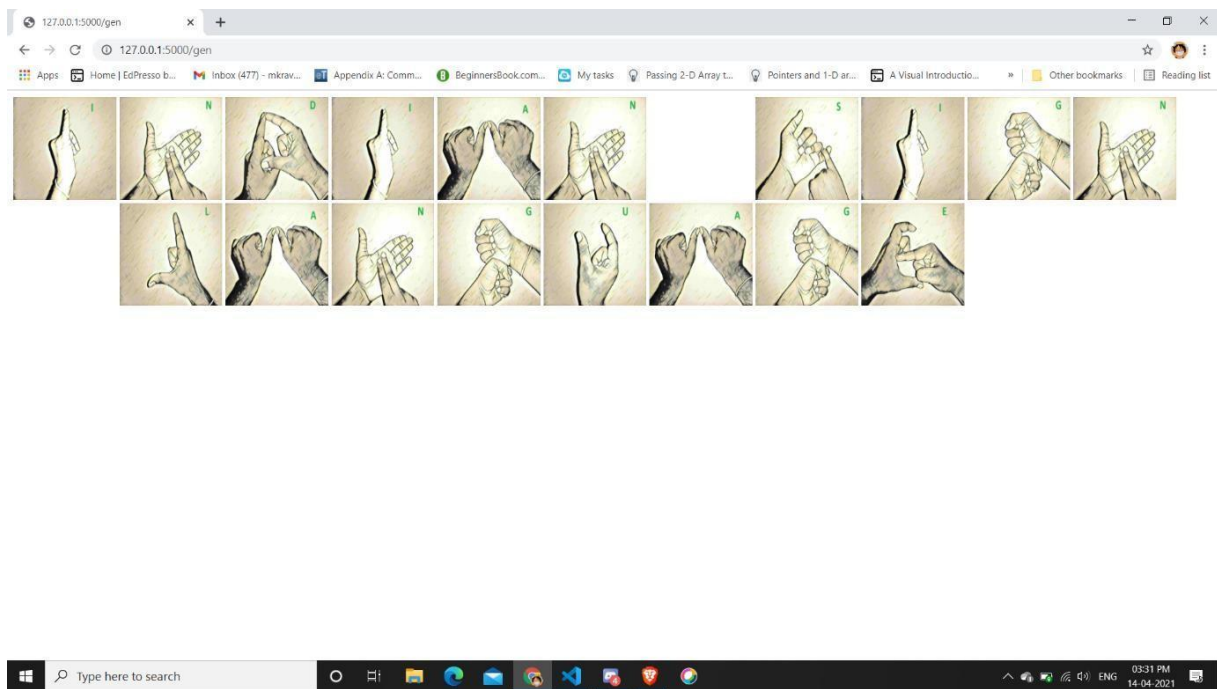


Fig 5.4: Corresponding Sign Language Output

The output of the web application is shown in the above image which is “Indian Sign Language” which was spoken as an input and then being recorded by the web application then it is being converted into the corresponding Indian Sign Language.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The conclusion of the project is that the produced web page can produce the output of hand sign based on the speech. The specialty about this project is that this helps to learn Indian Sign Language and also play a vital role in helping the deaf people. This will lead to the improvement in the quality of the sign language conversion.

Sign language serves as a primary mode of communication for deaf people. It is a major linguistic challenge that many researchers have taken up and built systems to facilitate the process of understanding the natural language employing sign language. This paper speaks about a system that is built mainly to take speech input in form of dialogues in English which are normally spoken and convert them to the corresponding sign language.

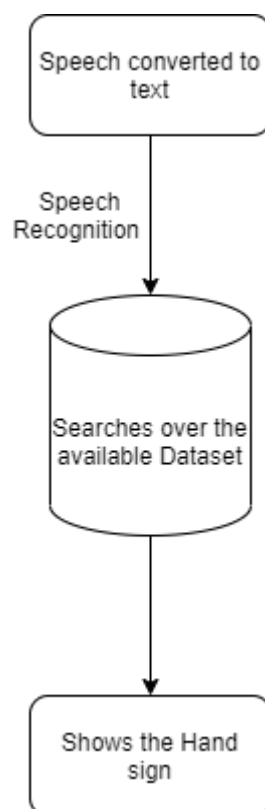


Fig 6.1 Flow of the Project

6.2 FUTURE WORK

This project helps to convert the speech into the corresponding text then from the text to sign language. This can be further developed or improvised by making the final output in the form of gif or in the form of a virtual avatar doing the corresponding signs. This will also help to improve the quality of the image and will be definitely helpful for the people who are physically challenged.

The output could be made into an full time application or an OS based application thus it can be more reliable and easy to use.

REFERENCES

- [1] Different approaches to convert speech into sign language Shahnaj Fatima; Ambuj Agarwal; Pooja Gupta, IEEE Conference 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)
- [2] G. Adithya V., Vinod P. R., Usha Gopalakrishnan, "Artificial Neural Network Based Method for Indian Sign Language Recognition", IEEE Conference on Information and Communication Technologies (ICT), 2013, pp. 1080-1085.
- [3] Speech into Sign Language Statistical Translation System for Deaf People B. Gallo, R. San-Segundo, J.M. Lucas, R. Barra, L.F. D'Haro, F. Fernández IEEE LATIN AMERICA TRANSACTIONS, VOL. 7, NO. 3, JULY 2009.
- [4] Vidal E., Casacuberta F, García P. "Gramatical Inference and Automatic Speech Recognition". New Advances and Trends in Speech Recognition and Coding (volume 147 of NATO-ASI Series F: Casacuberta and Vidal.
- [5] Yogeshwar I. Rokade, Prashant M. Jadav "Indian Sign Language recognition system, International Journal of engineering and Technology, vol. 9 No. 3S, pp.189-196, Jul. 2017.
- [6] R. Datta, J. Li, and J. Z. Wang, "Content-based image retrieval: approaches and trends of the new age," in Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval, pp. 253–262, ACM, Singapore, November 2005.
- [7] Z. Yu and W. Wang, "Learning DALTS for cross-modal retrieval," CAAI Transactions on Intelligence Technology, vol. 4, no. 1, pp. 9–16, 2019.
- [8] N. Ali, D. A. Mazhar, Z. Iqbal, R. Ashraf, J. Ahmed, and F. Zeeshan, "Content-based image retrieval based on late fusion of binary and local descriptors," International Journal of Computer Science and Information Security (IJCSIS), vol. 14, no. 11, 2016.

- [9] N. Ali, Image Retrieval Using Visual Image Features and Automatic Image Annotation, University of Engineering and Technology, Taxila, Pakistan, 2016.
- [10] B. Zafar, R. Ashraf, N. Ali et al., "Intelligent image classification-based on spatial weighted histograms of concentric circles," *Computer Science and Information Systems*, vol. 15, no. 3, pp. 615–633, 2018.
- [11] G. Qi, H. Wang, M. Haner, C. Weng, S. Chen, and Z. Zhu, "Convolutional neural network based detection and judgement of environmental obstacle in vehicle operation," *CAAI Transactions on Intelligence Technology*, vol. 4, no. 2, pp. 80–91, 2019.
- [12] U. Markowska-Kaczmar and H. Kwaśnicka, "Deep learning—a new era in bridging the speech language," in *Bridging the Semantic Gap in Image and Video Analysis*, pp. 123–159, Springer, Basel, Switzerland, 2018.

APPENDIX

A. Source Code

app.py

```
from flask import Flask, render_template, request, redirect
import speech_recognition as sr
import os

app = Flask(__name__)
arr=['_','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']

@app.route("/")
def index():
    return render_template("home.html")

@app.route("/text",methods=['POST'])
def text():
    if request.method == 'POST':
        a=request.form['inp']
        a=str(a)
        a=a.lower()
        print('nest')
        a=a.replace(' ','_')
        print(a)
        image_list = []
        for i in range(len(a)):
            if(a[i] in arr):
                image_list.append('static/data/letters/'+a[i]+' .jpg')
        print("images : ",image_list)
        return render_template('result.html', images=image_list)
    return render_template("home.html")

@app.route('/gen', methods=['POST', 'GET'])
```

```

def gen():
    try:
        path = 'static/temp/audio.wav'
        if request.method == "POST":
            f = request.files['audio_data']

            with open(path, 'wb') as audio:
                f.save(audio)
            print('file uploaded successfully')
        r = sr.Recognizer()
        text = sr.AudioFile(path)
        with text as source:
            audio = r.record(source)
        a=r.recognize_google(audio)
        a=a.lower()
        a=a.replace(' ','_')
        print("Text:"+a)
        image_list = []
        for i in range(len(a)):
            if(a[i] in arr):
                image_list.append('static/data/letters/'+a[i]+' .jpg')
        print(image_list)
        return render_template('result.html', images=image_list)
    except TypeError:
        print("refresh")

app.run(debug=True, use_debugger=False, use_reloader=False)

```

HTML FILE

Home.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>The Last page</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
      body {
        background-image: url('static/img/office.jpg');
        background-size: cover;
      }
    </style>

    <link rel="stylesheet" type="text/css" href="static/style/style.css">
  </head>
  <body>

    <header id="main-header">
      <h1 style="text-align: center; color: rgb(240, 6, 6);"> Sign Language Convertor </h1>
    </header>

    <div class="container" style="font-style: italic;">
      <form method="post" action="/text">
        <div style="padding: 10px;font-family: sans-serif;">
          <label for="inp1" style="color: lawngreen;">Type here</label><br>
          <input id="inp1" name="inp" placeholder="Enter the text here" type="text"
style="width: 250px; height: 50px;border-radius: 5px;border-color: burlywood;font-style:
italic;">
        </div>
        <div >
          <input type="submit" value="submit" style="border-width: 2px;font-style:
```

italic;background-color: firebrick;border-radius: 3px;">

</div>

</form>

<button id="recordButton">Record</button>

<button id="pauseButton" disabled>Pause</button>

<button id="stopButton" disabled>Stop</button>

<p id="formats">Format: start recording to see sample rate</p>

<p>Recordings:</p>

<ol id="recordingsList">

</div>

<!-- inserting these scripts at the end to be able to use all the elements in the DOM -->

<script

src="https://cdn.rawgit.com/mattdiamond/Recorderjs/08e7abd9/dist/recorder.js"></script

>

<script src="static/js/app.js"></script>

</body>

</html>

Result.html

```
<html>
  <body>
    {% for image in images %}
      
    {% endfor %}
  </body>
</html>
```

CSS FILE

Style.css

```
:root {
  --main-bg-color: #2b2c2e;
  --main-bg-color-fade-5: rgba(18, 0, 0, 0.51);
  --primary-color: #ff213c;
  --primary-shade-color: #d41c32;
  --secondary-color: #ba5ac7;
}

.mobile-nav {
  display: none;
}

.mb-1 {
  margin: 0;
  margin-bottom: 0.9em;
}

body, html {
  font-family: arial, sans-serif;
  background-color: var(--main-bg-color);
```

```
    color: #fff;
    margin: 0;

}

h1, h2, h3, h4, h5, h6 { color:
    var(--primary-color);
}

a {
    color: var(--primary-color);
    text-decoration: none;
    font-weight: bold;
}

h1 {
    font-size: 4em;
    color: rgba(192, 226, 41, 0.445);
}

h2 {
    font-size: 3.5em;
}

h3 {
    font-size: 2.8em;
    color: rgba(179, 222, 235, 0.281);
}

h4 {
    font-size: 2.4em;
}

h5 {
    font-size: 1.9em;
```

```
}
```

```
h6 {  
  font-size: 1.5em;  
}
```

```
button {  
  background-color: var(--primary-shade-color);  
  border-color: var(--primary-color);  
  color: #fff;  
  padding: 10px;  
  width: 100%;  
  font-weight: bold;  
  cursor: pointer;  
  margin-top: 10px;  
}
```

```
.margin-2 {  
  margin: 0.4em 0;  
}
```

```
.margin-3 {  
  margin: 0.8em 0;  
}
```

```
.container {  
  width: 900px;  
  min-width: 900px;  
  margin: 0 auto;  
}
```

```
.container-fluid {  
  width: 70%;  
  min-width: 30%;  
}
```

```
.hidden {  
    opacity: 0 !important;  
    height: 0 !important;  
    width: 0 !important;  
    overflow: hidden !important;  
}
```

```
/* navigation */  
#main-header, #main-header * {  
    transition: all 0.2s;  
    -o-transition: all 0.2s;  
    -moz-transition: all 0.2s;  
    -webkit-transition: all 0.2s;  
}
```

```
#main-header {  
    background-color: rgba(0, 0, 0, 0.1);  
    z-index: 3;  
}
```

```
#main-header.scrolled {  
    position: fixed;  
    background-color: #0d0a12;  
}
```

```
header > nav > ul {  
    margin: 0;  
    padding: 0;  
    display: flex;  
    list-style: none;  
    justify-content: center;  
}
```

```
header > nav > ul > li {  
    margin: 0 2% ;  
    position: relative;  
}
```

```
header > nav > ul > li.brand {  
    max-width: 9%;  
    margin: 0;  
}
```

```
header > nav > ul > li.brand >img {  
    width: 100%;  
}
```

```
header > nav > ul > li.brand > a {  
    padding: 0;  
}
```

```
header > nav > ul > li > a {  
    position: relative;  
    text-decoration: none;  
    font-size: 1.2em;  
    color: #fff;  
    display: block;  
    padding: 20px;  
}
```

```
header > nav > ul > li > a:before{  
    content: "";  
    height: 4px; width:  
    100%; position:  
    absolute;  
    background-color: #ff213c;  
    bottom: 0;  
    left: 0;
```

```
transform: scale3d(0, 1, 1);
transition: transform 0.1s;
}
```

```
header > nav > ul > li:hover > a{
  color:#ff213c;
}
```

```
header > nav .active > a::before {
  -webkit-transform: scale3d(1, 1, 1);
  transform: scale3d(1, 1, 1);
  -webkit-transition-timing-function: cubic-bezier(0.4, 0, 0.2, 1);
  transition-timing-function: cubic-bezier(0.4, 0, 0.2, 1);
  -webkit-transition-duration: 0.3s;
  transition-duration: 0.4s;
}
```

```
/* slider */
```

```
.slider > figure {
  margin: 0;
}
```

```
.slider > figure > img {
  width: 100%;
  max-height: 100vh;
}
```

```
.slider > .slide-caption {
  background-color: var(--main-bg-color);
  border-left: 5px solid var(--primary-color);
  color: rgba(211, 218, 206, 0.959);
}
```

```
    position: absolute;
    top: 15%;
    left: 4%;

    width: 25%;
    padding: 25px;
}
```

```
.slider > .slide-caption h1 {
    color: var(--primary-color);
    font-size: 4em;
    margin: 0;
}
```

```
.slider > .slide > .slide-caption h1 {
    color: var(--primary-color);
    font-size: 4em;
    margin: 0;
}
```

```
.slider > .slide > .slide-caption p {

}
```

```
/* home */
```

```
/* season */
```

```
.season > h2, .season p{
    margin: 0.8em 0;
}
```

```
.season > h2,
.season > h1{
    font-size: 1.8em;
```

```
}
```

```
.season > .episodes {  
  position: relative;  
  height: 450px;  
  overflow: hidden;  
}
```

```
#episode-number {  
  position: absolute;  
  right: 0;  
  background: rgba(0, 0, 0, 0.4);  
  padding: 10px;  
  z-index: 2;  
}
```

```
.season > .episodes,  
.season > .episodes > .episode {  
  transition: opacity 0.2s;  
  -o-transition: opacity 0.2s;  
  -moz-transition: opacity 0.2s;  
  -webkit-transition: opacity 0.2s;  
  margin: 0;  
  width: 100%;  
}
```

```
.season > .episodes > .episode {  
  position: absolute;  
  height: 450px;  
  z-index: 0;  
}
```

```
.season > .episodes > .episode > img {  
  width: 100%;
```

```
margin-top: -7%;  
}
```

```
.season > .episodes > .episode figcaption {  
  position: absolute;  
  bottom: 0;  
  background: rgba(0, 0, 0, 0.8);  
  width: 100%;  
  padding: 10px;  
  box-sizing: border-box;  
}
```

```
.season > .episodes > .episode > img {  
  transition: transform 0.5s;  
  -o-transition: transform 0.5s;  
  -moz-transition: transform 0.5s;  
  -webkit-transition: transform 0.5s;  
}
```

```
.season > .episodes > .episode > img:hover {  
  transform: scale(1.1);  
  -o-transform: scale(1.1);  
  -moz-transform: scale(1.1);  
  -webkit-transform: scale(1.1);  
}
```

```
.controls {  
  position: absolute;  
  width: 100%;  
  bottom: 50%;  
}
```

```
.controls > button {  
  background: #fff;  
  padding: 0.8em;
```

```
border: none;
cursor: pointer;
transition: all 0.1s ease-in-out;
font-weight: bold;
font-size: 16px;
width: 35px;
color: #000;

}

.controls > button:hover {
  color: #ff213c;
}

.controls .next {
  position: absolute;
  right: 0;
}

#episodes-group > .controls {
  z-index: 3;
}

#episodes-group > .episode {
  position: absolute;
  top: 0;
  opacity: 0;
  transition: opacity 0.5s;
}

#episodes-group > .episode:first-of-type {
  z-index: 2;
  opacity: 1;
}
```

```
/* characters */
#character-profile > .character {
    position: relative;
    margin-bottom: 250px;
}

#character-profile > .character > figure {
    overflow: hidden;
    height: 450px;
    margin: 0;
}

#character-profile > .character > figure > img {
    width: 100%;
    transition: transform 0.5s;
    -o-transition: transform 0.5s;
    -moz-transition: transform 0.5s;
    -webkit-transition: transform 0.5s;
}

#character-profile > .character > figure > img:hover { transform:
    scale(1.2);
    -o-transform: scale(1.2);
    -moz-transform: scale(1.2);
    -webkit-transform: scale(1.2);
}

#character-profile > .character > .character-bio {
    position: absolute;
    right: -50px;
    top: 45%;
    height: 250px;
```

```
background-color: #0d0a12;
padding: 25px;
width: 25%;

border-top: 10px solid var(--primary-color);
}

#character-profile > .character > .character-bio > h1 {
  font-size: 1.5em;
}

/* shop */
#webshop > .shop {
  display: flex;
  flex-wrap: wrap;
}

.shop > .shop-item {
  position: relative;
  width: 24%;
  background-color: #121410;
  margin: 2% 1%;
  padding: 2%;
}

.shop > .shop-item > .price-tag {
  background-color: var(--secondary-color);
  position: absolute;
  right: 10px;
  top: -40px;
  z-index: 2;

  height: 60px;
```

```
width: 48px;
padding: 10px;
}
```

```
.shop > .shop-item > .price-tag > .amount {
  position: absolute;
  bottom: 100px;
  height: 60px;
  width: 48px;
  padding: 10px
}
```

```
.shop > .shop-item figure {
  max-width: 250px;
  margin: 0 auto;
  transition: transform 0.2s;
  -o-transition: transform 0.2s;
  -moz-transition: transform 0.2s;
  -webkit-transition: transform 0.2s;
  z-index: 1;
  cursor: pointer;
}
```

```
.shop > .shop-item figure:hover {
  transform: scale(1.2);
  -o-transform: scale(1.2);
  -moz-transform: scale(1.2);
  -webkit-transform: scale(1.2);
}
```

```
.shop > .shop-item > figure > img {
  width: 100%;
}
```

```
.shop > .shop-item > .description {
```

```
position: relative;
font-style: italic;
text-align: center;
margin-top: 20px;
min-height: 75px;
z-index: 2;
}
```

```
.shop > .selected-shop-item {
width: 100%;
height: 100vh;
background-color: rgba(0, 0, 0, 0.5);
position: fixed;
top: 0;
left: 0;
z-index: 4;
transition: all 0.2s;
}
```

```
.shop > .selected-shop-item > .content {
display: grid;
grid-template-columns: 1fr 1fr;
max-width: 75%;
max-height: 60vh;
margin: 20vh auto;
background-color: #f3f3f3;
padding: 20px;
box-sizing: border-box;
color: #000;
position: relative;
}
```

```
.shop > .selected-shop-item > .content > .exit-button {
position: absolute;
right: 0px;
```

```
    cursor: pointer;
    background-color: #232323;
    text-align: center;
    padding: 10px;
    width: 40px;
    margin: 0;
    border: none;
}
```

```
.shop > .selected-shop-item > .content > figure {
    margin: 0 auto;
}
.shop > .selected-shop-item > .content > figure > img{
    width: 350px;
}
```

```
.shop > .selected-shop-item > .content > .details {
    display: grid;
    grid-template-rows: 1.4fr 1fr;
}
```

```
.shop > .selected-shop-item > .content > .details .item-name {
    margin: 0.2em 0;
}
```

```
.shop > .selected-shop-item > .content > .details input {
    padding: 2%;
    width: 100%;
    border: 1px solid darkgray;
    font-size: 1.4em;
    display: block;
    margin-top: 10%;
    box-sizing: border-box;
    margin-bottom: 2em;
}
```

```
}
```

```
.order-form {  
  display: grid;  
  grid-template-columns: 1fr 1fr;  
  grid-gap: 50px;  
}
```

```
.order-form article ul {  
  list-style: none;  
  margin: 0;  
  padding: 0;  
}
```

```
.order-form article ul li:before {  
  content: '\2714';  
  color: var(--primary-color);  
  margin-right: 1em;  
}
```

```
.order-form > form .field {  
  display: block;  
  margin: 10px 0;  
}
```

```
.order-form > form label {  
  display: block;  
  margin: 0.8em 0;  
}
```

```
.order-form > form input {  
  width: 100%;  
  padding: 20px 10px;  
  border-radius: 5px;  
  border: 2px solid var(--secondary-color);
```

```
    box-sizing: border-box;
}
```

```
footer {
    text-align: center;
    margin: 50px 0 20px;
    font-size: 0.8em;
    color: darkgray;
}
```

```
<article class="slide-caption center-left">
<h1> welcome </h1>
<p>Learn sign language </p>
</article>
```

JavaScript File

App.js

//webkitURL is depreciated

```
URL = window.URL || window.webkitURL;
```

```
var gumStream;           //stream from getUserMedia()
var rec;                  //Recorder.js object
var input;                //MediaStreamAudioSourceNode we'll be recording
```

// shim for AudioContext when it's not avb.

```
var AudioContext = window.AudioContext || window.webkitAudioContext;
var audioContext //audio context to help us record
```

```
var recordButton = document.getElementById("recordButton");
var stopButton = document.getElementById("stopButton");
var pauseButton = document.getElementById("pauseButton");
```

//add events to those 2 buttons

```
recordButton.addEventListener("click", startRecording);
```

```
stopButton.addEventListener("click", stopRecording);
pauseButton.addEventListener("click", pauseRecording);
```

```
function startRecording() {
```

```
    console.log("recordButton clicked");
```

```
    /*
```

```
        Simple constraints object, for more advanced audio features see
```

```
        https://addpipe.com/blog/audio-constraints-getusermedia/
```

```
    */
```

```
    var constraints = { audio: true, video:false }
```

```
    /*
```

```
        Disable the record button until we get a success or fail from getUserMedia()
```

```
    */
```

```
    recordButton.disabled = true;
```

```
    stopButton.disabled = false;
```

```
    pauseButton.disabled = false
```

```
    /*
```

```
        We're using the standard promise based getUserMedia()
```

```
        https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia
```

```
    */
```

```
    navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
```

```
        console.log("getUserMedia() success, stream created, initializing Recorder.js ...");
```

```
        /*
```

```
            create an audio context after getUserMedia is called
```

```
            sampleRate might change after getUserMedia is called, like it does on macOS
```

```
when recording through AirPods
```

```
            the sampleRate defaults to the one set in your OS for your playback device
```

```

*/
audioContext = new AudioContext();

//update the format
document.getElementById("formats").innerHTML="Format: 1 channel pcm @
"+audioContext.sampleRate/1000+"kHz"

/* assign to gumStream for later use */
gumStream = stream;

/* use the stream */
input = audioContext.createMediaStreamSource(stream);

/*
    Create the Recorder object and configure to record mono sound (1 channel) Recording
    2 channels will double the file size
*/
rec = new Recorder(input,{numChannels:1})

//start the recording process
rec.record()

console.log("Recording started");

}).catch(function(err) {
    //enable the record button if getUserMedia() fails
    recordButton.disabled = false;
    stopButton.disabled = true;
    pauseButton.disabled = true
});
}

function pauseRecording(){
    console.log("pauseButton clicked rec.recording=",rec.recording ); if
    (rec.recording){

```

```

        //pause
        rec.stop();
        pauseButton.innerHTML="Resume";
    }else{
        //resume
        rec.record()
        pauseButton.innerHTML="Pause";
    }
}

```

```

function stopRecording() {
    console.log("stopButton clicked");

    //disable the stop button, enable the record too allow for new recordings
    stopButton.disabled = true;
    recordButton.disabled = false;
    pauseButton.disabled = true;

    //reset button just in case the recording is stopped while paused
    pauseButton.innerHTML="Pause";

    //tell the recorder to stop the recording
    rec.stop();

    //stop microphone access
    gumStream.getAudioTracks()[0].stop();

    //create the wav blob and pass it on to createDownloadLink
    rec.exportWAV(createDownloadLink);
}

```

```

function createDownloadLink(blob) {

    var url = URL.createObjectURL(blob);

```

```

var au = document.createElement('audio');
var li = document.createElement('li');
var link = document.createElement('a');

//name of .wav file to use during upload and download (without extension) var
filename = new Date().toISOString();

//add controls to the <audio> element
au.controls = true;
au.src = url;

//save to disk link
link.href = url;
link.download = filename+".wav"; //download forces the browser to download the file
using the filename
link.innerHTML = "Save to disk";

//add the new audio element to li
li.appendChild(au);

//add the filename to the li
li.appendChild(document.createTextNode(filename+".wav "))

//add the save to disk link to li
li.appendChild(link);

//upload link
var upload = document.createElement('a');
upload.href="/gen";
upload.innerHTML = "Upload";
upload.addEventListener("click", function(event){
    var xhr=new XMLHttpRequest();
    xhr.onload=function(e) {
        if(this.readyState === 4) {
            console.log("Server returned: ",e.target.responseText);

```

```
    }  
};  
var fd=new FormData();  
fd.append("audio_data",blob, filename);  
xhr.open("POST","/gen",true);  
xhr.send(fd);  
})  
li.appendChild(document.createTextNode (" "))//add a space in between  
li.appendChild(upload)//add the upload link to li  
  
//add the li element to the ol  
recordingsList.appendChild(li);  
}
```