

Final Report

A Real Time Communication System For Specially Abled

Team Members:

G S AKSHAYA

K ANITHA

R KAVYAPRIYA

SWASTHIKA VENKATARAMAN

Table Of Contents

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

- 6.1 Sprint Planning & Estimation
- 6.2 Sprint Delivery Schedule
- 6.3 Reports from JIRA

7. CODING & SOLUTIONING

- 7.1 Libraries to be installed
- 7.2 Real time sign to speech
- 7.3 Facial Emotion Detection
- 7.4 Language Customization
- 7.5 Real time speech to text

8. TESTING

- 8.1 Test Cases
- 8.2 User Acceptance Testing

9. RESULTS

- 9.1 Performance Metrics

10.ADVANTAGES & DISADVANTAGES

11.CONCLUSION

12.FUTURE SCOPE

13.APPENDIX

13.1 Source Code

13.2 GitHub & Project Demo Link

1. Introduction

1.1 Project Overview

People with impairments are a part of our society. Although technology is constantly evolving, little is being done to improve the lives of these people. The ability to communicate with a deaf-mute person has always been difficult. It is quite challenging for silent persons to communicate with non-mute people because hand sign language is not taught to the general public. It might be quite challenging for them to communicate at times of crisis. In circumstances where other modes of communication, like speech, are not possible, the human hand has remained a common alternative for information transmission. To have proper communication between a normal person and a handicapped person in any language, a voice conversion system with hand gesture recognition and translation will be very helpful.

1.2 Purpose

The project intends to create a system that can translate speech into specified sign language for the deaf and dumb as well as translate sign language into a human hearing voice in the desired language to communicate a message to normal people. A convolution neural network is being used to build a model that is trained on various hand motions. Based on this model, an app is created. With the help of this app, persons who are deaf or dumb can communicate using signs that are translated into speech and human-understandable words.

2. Literature Survey

2.1 Existing Problem

In our society, we have people with disabilities. The technology is developing day by day but no significant developments are undertaken for the betterment of these people. Communications between deaf-mute and a normal person has always been a challenging task. It is complicated for mute people to convey their message to normal people. Since normal people are not trained on hand sign language. In emergency times conveying their message is very difficult. The human hand has remained a popular choice to convey information in situations where other forms like speech cannot be used. Voice Conversion System with Hand Gesture Recognition and translation will be handy to have a proper conversation between a normal person and an impaired person in any language.

2.2 References

Design of Communication Interpreter for Deaf and Dumb Person was published by Pallavi Verma (Electrical and Electronics Department, Amity University, Greater Noida, Uttar Pradesh, India), Shimi S. L (Assistant Professor, NITTTR, Chandigarh, India), Richa Priyadarshani (Electrical and Electronics Department, Amity University, Greater Noida, Uttar Pradesh, India).
International Journal of Science and Research (IJSR) · Jan 2013

Development of full duplex intelligent communication system for deaf and dumb people was published in the year January 2017
DOI:10.1109/CONFLUENCE.2017.7943247

At 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence (Confluence) by Surbhi Rathi Department of Information Technology, Yeshwantrao Chavan College of Engineering Nagpur, India and Ujwalla Gawande, Department of Information Technology Yeshwantrao Chavan College of Engineering Nagpur, India.

A Review Paper on Sign Language Recognition for The Deaf and Dumb published by R Rumana(B.E Graduate(IV year), Department of Computer Science and Engineering, SCSVMV, Kanchipuram) , Reddygari Sandhya Rani(B.E Graduate(IV year), Department of Computer Science and Engineering, SCSVMV, Kanchipuram) , Mrs. R. Prema(Assistant Professor, Department of Computer Science and Engineering, SCSVMV, Kanchipuram).

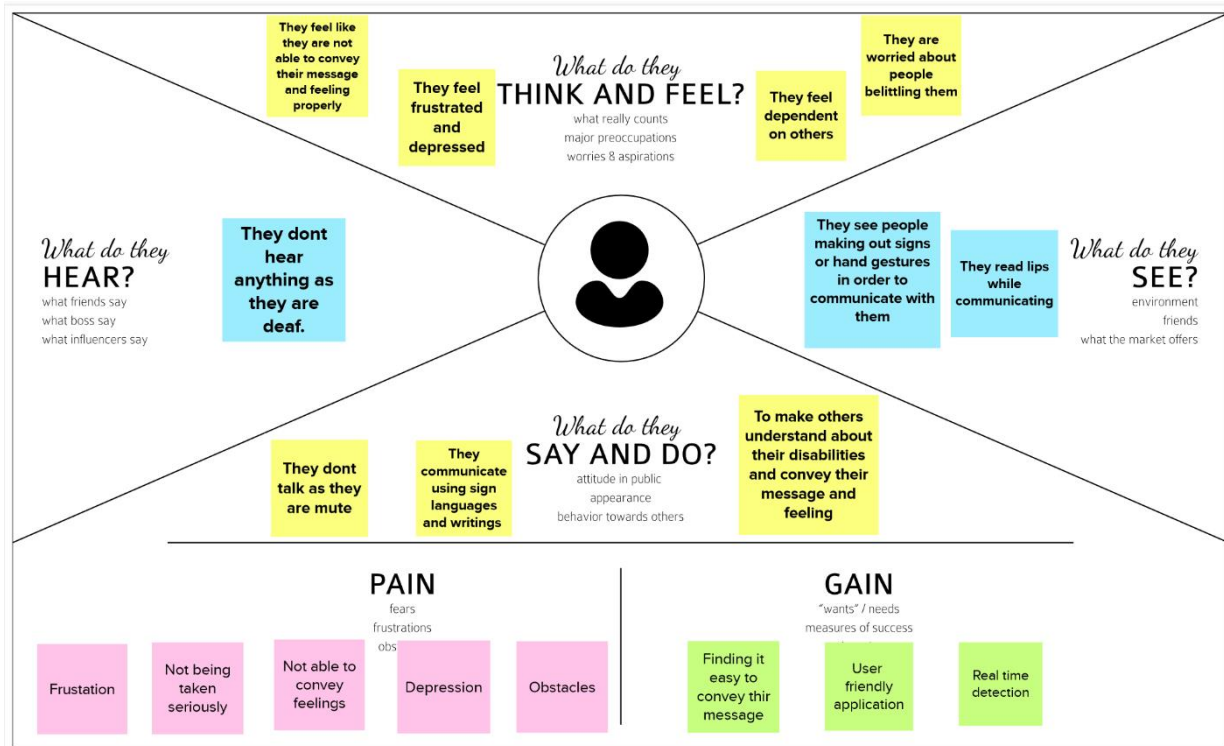
Published (First Online): 01-11-2021

2.3 Problem Definition Statement

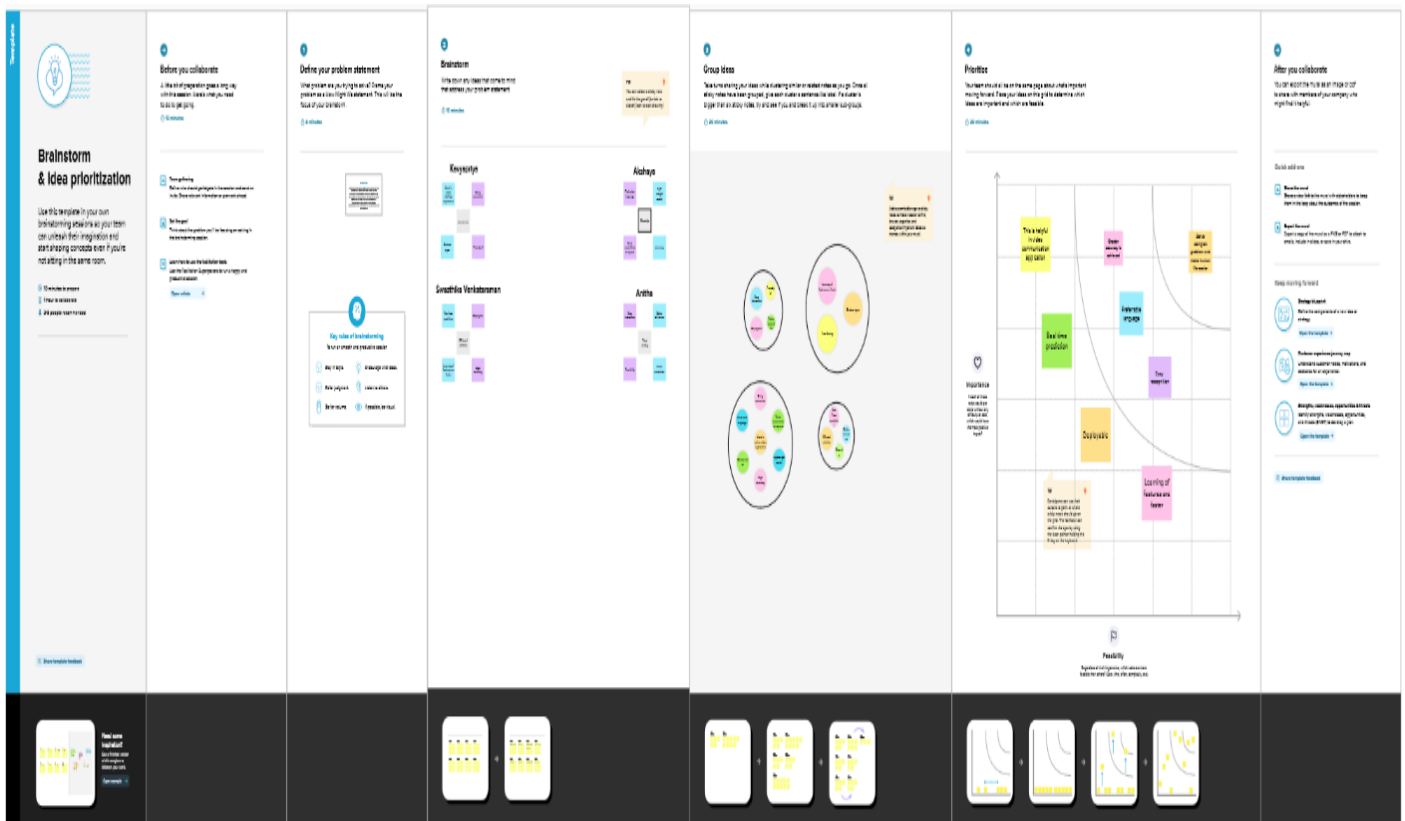
People with disabilities are a part of our society. Even though technology is constantly evolving, little is being done to improve the lives of these people. Communication with a deaf-mute person has always been difficult. Because hand sign language is not taught to the general public, it can be difficult for silent people to communicate with non-mute people. In times of crisis, they may find it difficult to communicate. When other modes of communication, such as speech, are unavailable, the human hand has remained a popular method of information transmission. A voice conversion system with hand gesture recognition and translation will be very helpful in establishing proper communication between a normal person and a handicapped person in any language.

3. Ideation and Proposed Solution

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming

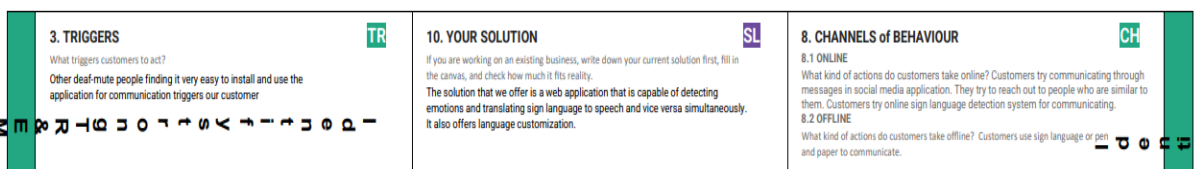
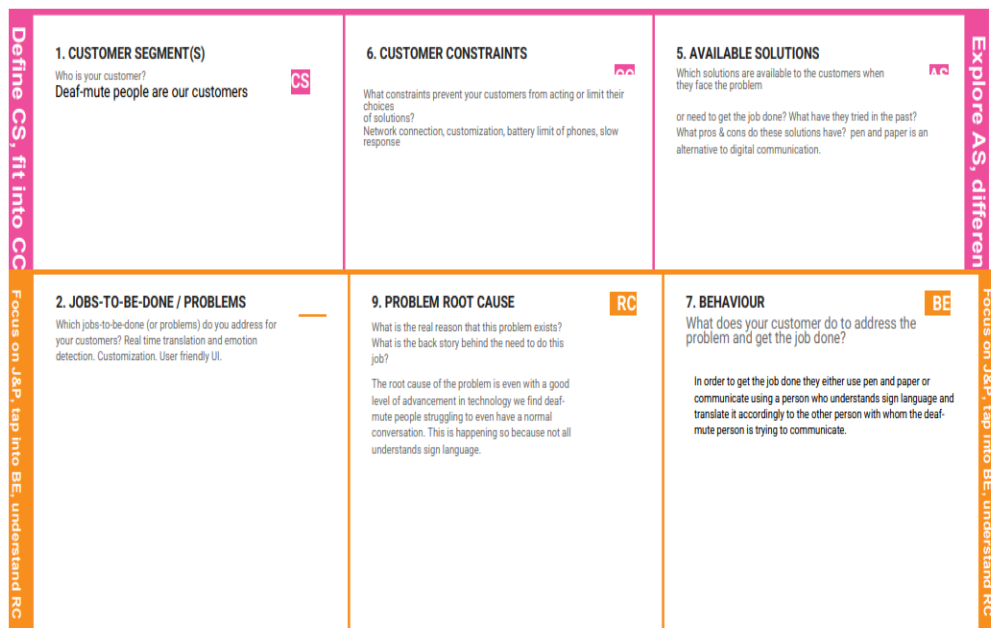


3.3 Proposed Solution

S No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In our society, we have people with disabilities. The technology is developing day by day but no significant developments are undertaken for the betterment of these people. Communications between deaf-mute and a normal person has always been a challenging task. It is very difficult for mute people to convey their message to normal people. Since normal people are not trained on hand sign language. In emergency times conveying their message is very difficult. The human hand has remained a popular choice to convey information in situations where other forms like speech cannot be used. Voice Conversion System with Hand Gesture Recognition and translation will be very useful to have a proper conversation between a normal person and an impaired person in any language.
2.	Idea / Solution description	The project aims to develop a system that converts the sign language into a human hearing voice in the desired language to convey a message to normal people, as well as convert speech into understandable sign language for the deaf and dumb. We are making use of a convolution neural network to create a model that is trained on different hand gestures. An app is built which uses this model. This app enables deaf and dumb people to convey their information using signs which get converted to human-understandable language and speech is given as output.
3.	Novelty / Uniqueness	<ul style="list-style-type: none"> • Facial Emotion Detection • Language customization • User-friendly interface. • Greater accuracy.
4.	Social Impact / Customer Satisfaction	The proposed solution is keen on providing a friendly user interface and user experience. User Interface (UI) is aimed to be developed in such a that way that it can be very handy and easy to learn. The system is also aimed to be light weight which would make the system provide faster and accurate results and hence it provides a better User Experience (UX).

5.	Business Model (Revenue Model)	The proposed solutions help to ease the communication between deaf and dumb people and normal people. The customization and emotion detection feature can make it lot more reliable. Hence, the solution has wide usability and requirement.
6.	Scalability of the Solution	This proposed solution is highly extensible in terms of the features that is been offered by the system. It can be seen as a highly improvised and light weight model when compared to the existing systems. The system can further be scaled in such a way that enables tasks being assigned and completed in system through gestures.

3.4 Proposed Solution Fit



4. Requirement Analysis

4.1 Functional Requirement

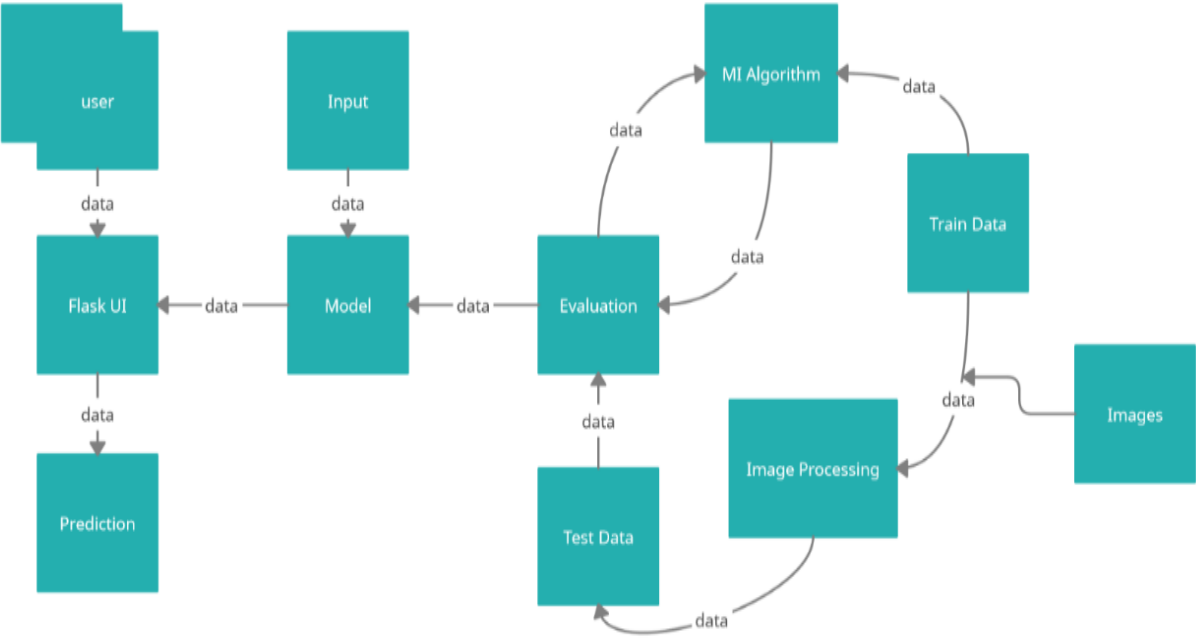
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Language customization	The user performs language customization.
FR-2	User Options	The user either chooses to convert speech to sign language and sign language to speech.
FR-3	Test Inputs	The real time video and audio data is collected and fed into the machine learning model.
FR-4 x	Result	The conversion will take place simultaneously and will be displayed on the screen.

4.2 Non-Functional Requirement

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The user will have access to all the resources present in that website.
NFR-2	Security	User information is protected.
NFR-3	Reliability	It offers accurate results.
NFR-4	Performance	The web application makes use of light weight model hence the result will be accurate and fast.
NFR-5	Availability	The web application can be accessed 24/7 from anywhere when connected to the internet.
NFR-6	Scalability	The trained ML model can provide accurate results whenever the size of the dataset and the number of users is extended.

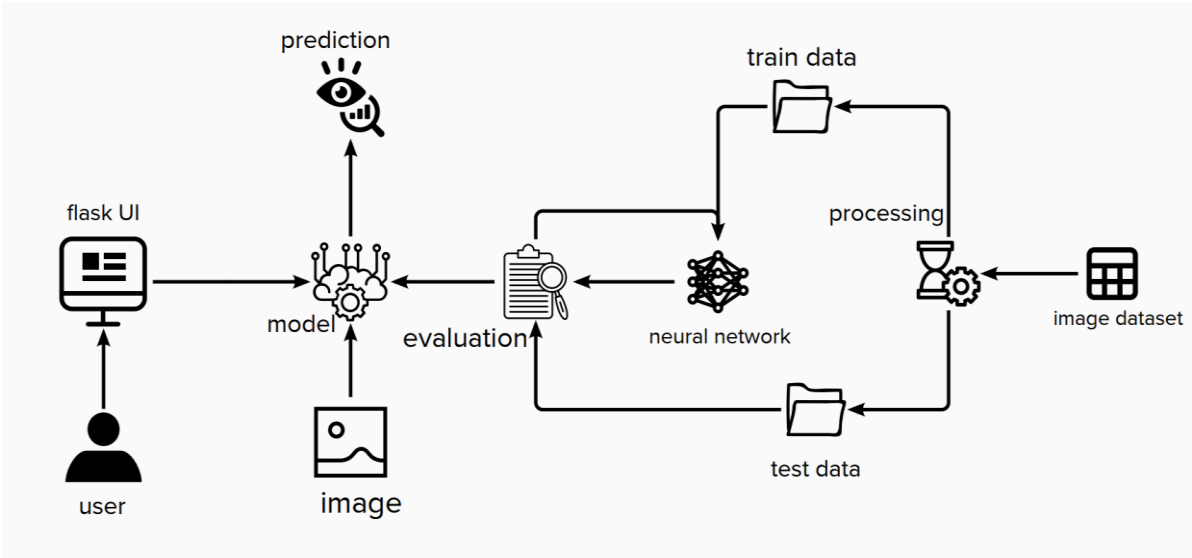
5. Project Design

5.1 Dataflow Diagram

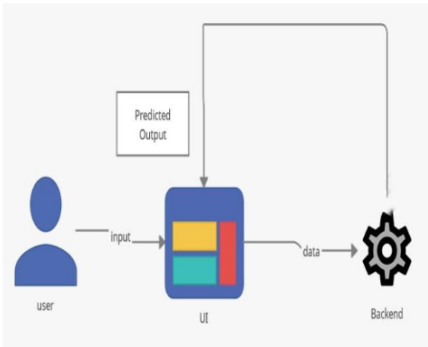


5.2 Solution and Technical Architecture

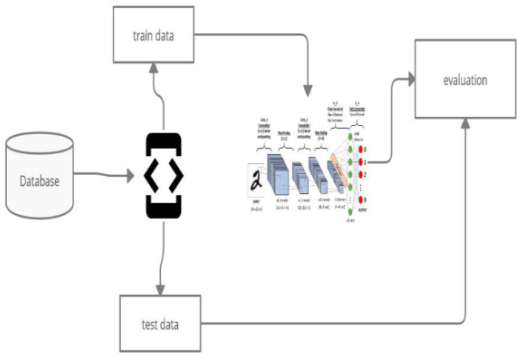
Solution Architecture



Technical Architecture



DEPLOYMENT



TRAINING AND EVALUATION

5.3 User Story

User	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Uploading the real time data.	USN-1	The user will be presented with two options. 1. Speech to sign language conversion. 2. Sign language to speech conversion.	They can access the portal	High	Sprint-1
		USN-2	Language selection	They can access the portal	Low	Sprint-1
		USN-3	The deaf-mute person will choose the speech to sign language conversion which would take them into a portal that collects the real time data (sign language recognition) and converts it into speech simultaneously.	Video processing	High	Sprint-2
		USN-4	Emotion detection	Video processing	Medium	Sprint-1
		USN-5	Normal person would choose speech to sign language which would take them into a portal where their speech is converted into sign language simultaneously.	Video and audio processing	High	Sprint-1

6. Project Planning and Scheduling

6.1 Sprint Planning and Estimation

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Choose Available Options	USN-1	The user will be presented with two options. 1. Speech to sign language conversion. 2. Sign language to speech conversion	1	Low	G.S Akshaya Swastika Venkataraman R Kavyapriya K Anitha
Sprint-1	Dashboard	USN-2	Go to dashboard and see the available features	1	Low	G.S Akshaya Swastika Venkataraman R Kavyapriya K Anitha
Sprint-2	Language selection	USN-3	The user can select any one of the available options according to their requirement.	2	Medium	G.S Akshaya Swastika Venkataraman R Kavyapriya K Anitha
Sprint-3	Convert from one language to another	USN-4	The deaf-mute person will choose the speech to sign language conversion which would take them into a portal that collects the real time data (sign language recognition) and converts it into speech simultaneously	2	High	G.S Akshaya Swastika Venkataraman R Kavyapriya K Anitha

Sprint	Functional Requirement (Epic)	User Story Number	User Story Task	Story Points	Priority	Team Members
Sprint-4	Emotion detection	USN-5	By processing the video it detects the emotion of the user.	2	High	G.S Akshaya Swastika Venkataraman R Kavyapriya K Anitha
Sprint-1	Exit	USN-6	Click exit button to exit from the application	1	Low	G.S Akshaya Swastika Venkataraman R Kavyapriya K Anitha

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

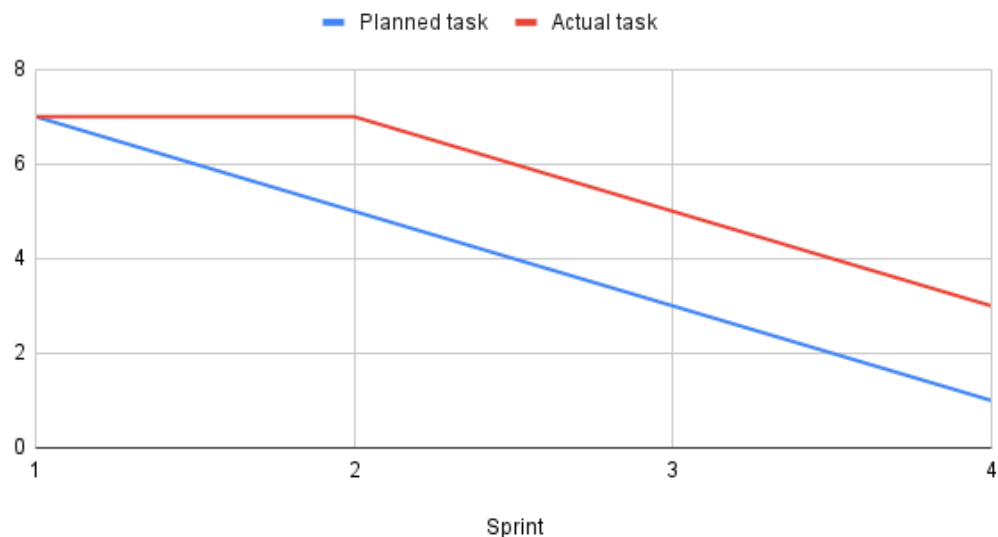
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

Burndown Chart:

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time

Planned task and Actual task



6.2 Reports From JIRA

11/17/22, 9:44 PM

SmartBear Test Management

Coverage Report

Coverage	Test Cases
No Coverage10	RCSFSA-T1 APPROVED Welcomepage_TC_001
	RCSFSA-T2 APPROVED Welcomepage_TC_002
	RCSFSA-T3 APPROVED Language Selection_TC_001
	RCSFSA-T4 APPROVED Sign to speech_TC_001
	RCSFSA-T5 APPROVED Sign to speech_TC_002
	RCSFSA-T6 APPROVED Sign to speech_TC_003
	RCSFSA-T7 APPROVED Speech to sign_TC_001
	RCSFSA-T8 APPROVED Speech to sign_TC_002
	RCSFSA-T9 APPROVED Speech to sign_TC_003
	RCSFSA-T10 APPROVED Speech to sign_TC_004

Displaying (1 of 1)

<https://swasthikavenkataraman.atlassian.net/projects/RCSFSA?selectedItem=com.atlassian.plugins.atlassian-connect-plugin:com.kanoah.test-manag...> 1/1

11/17/22, 9:45 PM

SmartBear Test Management

Traceability Report

Coverage	Test Cases	Test Execution Results	Issues
No Coverage10	RCSFSA-T1 APPROVED Welcomepage_TC_0011	PASS 0 Executed on: 17/Nov/22 8:50 pm Environment: - Executed by: Swasthika Venkataraman	None
	RCSFSA-T2 APPROVED Welcomepage_TC_0021	PASS 0 Executed on: 17/Nov/22 9:41 pm Environment: - Executed by: Swasthika Venkataraman	None
	RCSFSA-T3 APPROVED Language Selection_TC_0011	PASS 0 Executed on: 17/Nov/22 8:54 pm Environment: - Executed by: Swasthika Venkataraman	None
	RCSFSA-T4 APPROVED Sign to speech_TC_0011	PASS 0 Executed on: 17/Nov/22 9:44 pm Environment: - Executed by: Swasthika Venkataraman	None
	RCSFSA-T5 APPROVED Sign to speech_TC_0021	PASS 0 Executed on: 17/Nov/22 9:01 pm Environment: - Executed by: Swasthika Venkataraman	None
	RCSFSA-T6 APPROVED Sign to speech_TC_0031	PASS 0 Executed on: 17/Nov/22 9:04 pm Environment: - Executed by: Swasthika Venkataraman	None
	RCSFSA-T7 APPROVED Speech to sign_TC_0011	PASS 0 Executed on: 17/Nov/22 9:07 pm Environment: - Executed by: Swasthika Venkataraman	None

https://swasthikavenkataraman.atlassian.net/projects/RCSFSA?selectedItem=com.atlassian.plugins.atlassian-connect-plugin:com.kanoah.test-manag... 1/2

Coverage	Test Cases	Test Execution Results	Issues
	RCSFSA-T8 APPROVED Speech to sign_TC_OO2 1	PASS 0 Executed on: 17/Nov/22 9:12 pm Environment: - Executed by: Swasthika Venkataraman	None
	RCSFSA-T9 APPROVED Speech to sign_TC_OO3 1	PASS 0 Executed on: 17/Nov/22 9:18 pm Environment: - Executed by: Swasthika Venkataraman	None
	RCSFSA-T10 APPROVED Speech to sign_TC_OO4 1	PASS 0 Executed on: 17/Nov/22 9:24 pm Environment: - Executed by: Swasthika Venkataraman	None

Displaying (1 of 1)

Traceability matrix

Coverage	Test Cases									
	RCSFSA-T1 - Welcomepage_...	RCSFSA-T2 - Welcomepage_...	RCSFSA-T3 - Language Selec...	RCSFSA-T4 - Sign to speech_...	RCSFSA-T5 - Sign to speech_...	RCSFSA-T6 - Sign to speech_...	RCSFSA-T7 - Speech to sign_...	RCSFSA-T8 - Speech to sign_...	RCSFSA-T9 - Speech to sign_...	RCSFSA-T10 - Speech to sign...
No Coverage										

Displaying (1 of 1)

Last test execution: Pass

Traceability Tree

Traceability	Summary
No Coverage	
└─ Covered by Test Case RCSFSA-T1	Welcomepage_TC_001
└─ Executed on 17/Nov/22 8:50 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T2	Welcomepage_TC_OO2
└─ Executed on 17/Nov/22 9:41 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T3	Language Selection_TC_001
└─ Executed on 17/Nov/22 8:54 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T4	Sign to speech_TC_OO1
└─ Executed on 17/Nov/22 9:44 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T5	Sign to speech_TC_OO2
└─ Executed on 17/Nov/22 9:01 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T6	Sign to speech_TC_OO3
└─ Executed on 17/Nov/22 9:04 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T7	Speech to sign_TC_OO1
└─ Executed on 17/Nov/22 9:07 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T8	Speech to sign_TC_OO2
└─ Executed on 17/Nov/22 9:12 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T9	Speech to sign_TC_OO3
└─ Executed on 17/Nov/22 9:18 pm	PASS Executed by Swasthika Venkataraman
└─ Covered by Test Case RCSFSA-T10	Speech to sign_TC_OO4
└─ Executed on 17/Nov/22 9:24 pm	PASS Executed by Swasthika Venkataraman

Displaying (1 of 1)

7. Coding and Solutioning

7.1 Libraries to be installed

```
pip install fer  
pip install flask  
pip install cv2  
pip install numpy  
pip install keras  
pip install tensorflow  
pip install cvzone  
pip install pyttsx3  
pip install scikit-image
```

7.2 Real time sign to speech

Sign language is generally used by the people who are unable to speak, for communication. Most people will not be able to understand the Universal Sign Language (unless they have learnt it) and due to this lack of knowledge about the language, it is very difficult for them to communicate with mute people. A device that helps to bridge a gap between mute persons and other people forms the crux of this project. Our system makes use of a model build using CNN that is capable of detection sign languages real time.

7.3 Facial Emotion Detection

Our system makes use of the FER model. Facial Emotion Recognition (commonly known as FER) is one of the most researched fields of computer vision till date and is still in continuous evaluation and improvement. The model is a convolutional neural network with weights saved to HDF5 file in the data folder relative to the module's path. It can be overridden by injecting it into the `FER()` constructor during instantiation with the `emotion_model` parameter.

7.4 Language Customization

Google Translate is a free multilingual machine translation service. It can translate the Website's text content from one language to another. It offers a huge list of languages to translate and has an efficient, reliable and easy way to translate the webpage in whatever language the user wants. It supports over 100 languages. Use this website translator to convert webpages into your choice of language.

7.5 Real time speech to text

With the Web Speech API, we can recognize speech using JavaScript. It is super easy to recognize speech in a browser using JavaScript and then getting the text from the speech to use as user input. We use the **SpeechRecognition** object to convert the speech into text and then display the text on the screen. Our system is capable of doing this over real time. It is capable of recognizing any languages in which the user is trying to communicate. But the support for this API is limited to the **Chrome browser only**. So if you are viewing this example in some other browser, the live example below might not work.

8. Testing

8.1 Test Cases

- Verify if user can see the options when user clicks the URL
- Verify if the UI elements are getting displayed properly
- Verify if the user can choose any languages
- Verify if the user is getting redirected to the sign to speech page
- Verify if the application can convert the sign to speech
- Verify if the user can exit the sign to speech page
- Verify if the user is getting redirected to the speech to sign page
- Verify if the UI elements are being displayed
- Verify if the application can convert speech to text on clicking voice to text button.
- Verify if the user can exit the speech to sign page.

8.2 UAT Testing

1. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

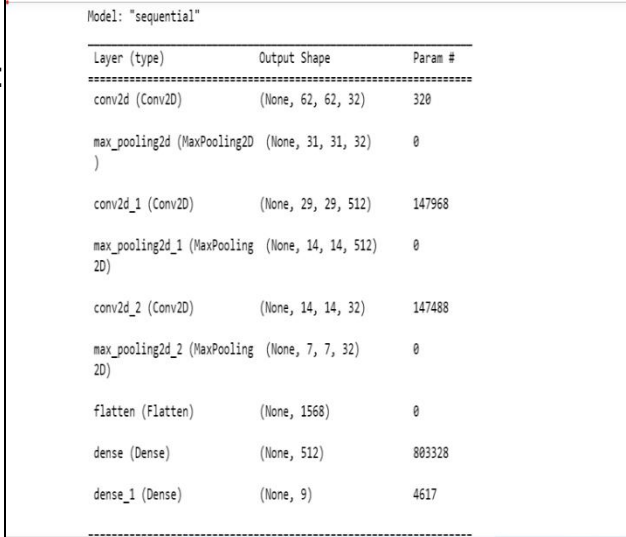
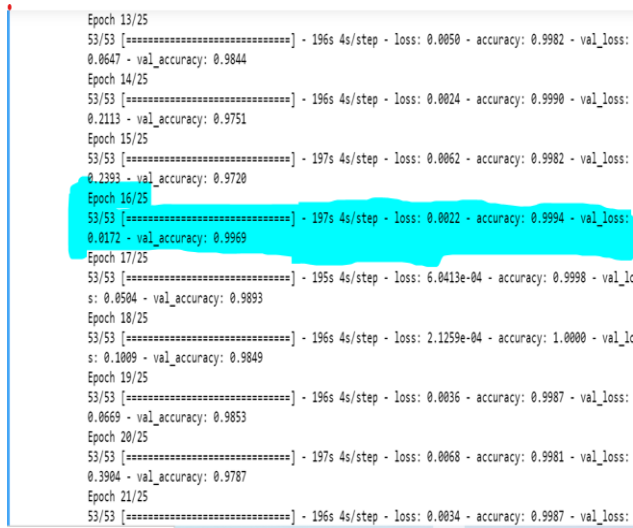
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	11	7	4	2	24
Duplicate	1	0	2	0	3
External	2	3	2	1	8
Fixed	10	5	3	14	32
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	1	0	0	0	1
Totals	25	15	13	18	71

2. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	15	0	0	15
Security	2	0	0	2
Outsource Shipping	2	0	0	2
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

8.3 Performance Testing

S.NO	Parameter	Values	Screenshot
1.	Model Summary	Total params: 1,103,721 Trainable params: 1,103,721 Non-trainable params: 0	 <pre> Model: "sequential" Layer (type) Output Shape Param # ----- conv2d (Conv2D) (None, 62, 62, 32) 320 max_pooling2d (MaxPooling2D) (None, 31, 31, 32) 0 conv2d_1 (Conv2D) (None, 29, 29, 512) 147968 max_pooling2d_1 (MaxPooling2D) (None, 14, 14, 512) 0 conv2d_2 (Conv2D) (None, 14, 14, 32) 147488 max_pooling2d_2 (MaxPooling2D) (None, 7, 7, 32) 0 flatten (Flatten) (None, 1568) 0 dense (Dense) (None, 512) 803328 dense_1 (Dense) (None, 9) 4617 </pre>
2.	Accuracy	Training Accuracy - 0.9994 Validation Accuracy -0.9969	 <pre> Epoch 13/25 53/53 [=====] - 196s 4s/step - loss: 0.0050 - accuracy: 0.9982 - val_loss: 0.0647 - val_accuracy: 0.9844 Epoch 14/25 53/53 [=====] - 196s 4s/step - loss: 0.0024 - accuracy: 0.9990 - val_loss: 0.2113 - val_accuracy: 0.9751 Epoch 15/25 53/53 [=====] - 197s 4s/step - loss: 0.0062 - accuracy: 0.9982 - val_loss: 0.2393 - val_accuracy: 0.9720 Epoch 16/25 53/53 [=====] - 197s 4s/step - loss: 0.0022 - accuracy: 0.9994 - val_loss: 0.0172 - val_accuracy: 0.9969 Epoch 17/25 53/53 [=====] - 195s 4s/step - loss: 6.0413e-04 - accuracy: 0.9998 - val_loss: 0.0504 - val_accuracy: 0.9893 Epoch 18/25 53/53 [=====] - 196s 4s/step - loss: 2.1259e-04 - accuracy: 1.0000 - val_loss: 0.1009 - val_accuracy: 0.9849 Epoch 19/25 53/53 [=====] - 196s 4s/step - loss: 0.0036 - accuracy: 0.9987 - val_loss: 0.0669 - val_accuracy: 0.9853 Epoch 20/25 53/53 [=====] - 197s 4s/step - loss: 0.0068 - accuracy: 0.9981 - val_loss: 0.3904 - val_accuracy: 0.9787 Epoch 21/25 53/53 [=====] - 196s 4s/step - loss: 0.0034 - accuracy: 0.9987 - val_loss: </pre>

[13:44:32.751] http://127.0.0.1:5000/							
Timing type <input type="radio"/> Relative <input checked="" type="radio"/> Independent							
Time	Response	Req. Size	Resp. Size	Analysis	Total time	Timing	
13:44:32.751	200	738	1626		22 ms	±	
GET http://127.0.0.1:5000/							
13:44:32.796	200	430	115881		10 ms	±	
GET http://gc.kis.v2.scr.kaspersky-labs.com/FD126C42-EBFA-4E12-B309-BB3FDD723AC1/main.							
13:44:32.796	301	349	481		223 ms	±	
GET http://translate.google.com/translate_a/element.js							
13:44:32.797	200	36	323		9 ms	±	
GET http://127.0.0.1:5000/static/css/index.css							
13:44:32.826	101	733	—		16153 ms	±	
GET ws://gc.kis.v2.scr.kaspersky-labs.com/7D8B79A2-8974-4D7B-A76A-F4F29624C06B4Ww5w3wj4_-aTY2acSeEt6b-yxd-EUnxd1WUJ							
13:44:33.019	200	—	—		686 ms	±	
GET https://translate.google.com/translate_a/element.js							
13:44:33.718	200	—	—		0 ms	—	
GET https://translate.googleapis.com/translate_static/css/translateelement.css							
13:44:33.719	200	—	—		18 ms	±	
GET https://translate.googleapis.com/_/translate_http/_js/k=translate_http.tr.en_GB.jUY4_WDT6tY.O/d=							
13:44:33.722	200	—	—		11 ms	±	
GET https://fonts.googleapis.com/css2							
13:44:33.739	200	—	—		6 ms	±	
GET https://image.shutterstock.com/image-vector/deaf-sign-language-welcome-line-260nw-1585429861							
13:44:33.739	200	—	—		6 ms	±	
GET https://fonts.gstatic.com/s/roboto/v30/KFOlCnqEu92Fr1MmYUtlBbc4.woff2							
13:44:33.739	200	—	—		6 ms	±	
GET https://fonts.gstatic.com/s/oxygen/v15/2sDcZG1Wl4LcnbuCNWqzaGW5.woff2							

10. Advantages and Disadvantages

Advantages:

- Real time sign to speech detection.
- Model provides good accuracy.
- Real time facial emotion detection.
- Language Customization.
- Real time speech to text conversion.
- Friendly UI
- Data privacy

Disadvantages:

- At times the website may lag.
- Model is not tested on a wide set of data set, having all the signs.
- Sign language customization feature is not available.
- User cannot take notes while using the app.
- User cannot make calls using the app.
- Speech recognition works only on google chrome.

11. Conclusion

Communication is crucial for self-expression. Additionally, it meets one's necessities. Effective communication is necessary for career advancement. Effective communication skills can make your personal life easier and improve your interactions with others by facilitating mutual understanding. A system that translates speech into acceptable sign language for the deaf and dumb has been developed as part of our project. It also translates sign language into a human hearing voice to communicate with average people. A convolution neural network has been used to build a model that is trained on various hand motions. Utilizing this concept, an app is created. Through the use of signs that are translated into speech and human-understandable English, this software aids deaf and dumb individuals to communicate easily.

12. Future Scope

The following are the features that can be added in our application:

- A communication app can be built with the same set of features. The user can choose the appropriate mode (speech to sign or sign to speech) and accordingly the real time detection would take place on both the end users' application.
- The accuracy of the model shall be increased.
- Customization of languages shall be added.
- Users shall be allowed to write notes while on call.
- Customization of signs can also be added as a feature.

13. Appendix

Source Code

Model Building

```
import cv2

import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import numpy as np

from keras.models import Sequential

import matplotlib.pyplot as plt

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Conv2D, MaxPool2D

from keras_preprocessing.image import ImageDataGenerator

test_path = 'Dataset/test_set'

train_path = 'Dataset/training_set'

train=ImageDataGenerator(rescale=1./255,zoom_range=0.2,shear_range=0.2,horizontal_flip=True)

test=ImageDataGenerator(rescale=1./255)

train_batches = train.flow_from_directory(directory=train_path, target_size=(64,64),
class_mode='categorical', batch_size=300,shuffle=True,color_mode="grayscale")

test_batches = test.flow_from_directory(directory=test_path, target_size=(64,64),
class_mode='categorical', batch_size=300, shuffle=True,color_mode="grayscale")

model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(64,64,1)))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(512, (3, 3), padding="valid"))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(32, (3, 3), padding="same"))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(512,activation = "relu"))

model.add(Dense(9,activation = "softmax"))
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(train_batches, batch_size=32, validation_data=test_batches, epochs=25)

model.save('model.h5')
```

Model Testing

```
import keras
from keras.models import load_model
import cv2
import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
val=['A','B','C','D','E','F','G','H','I']

model=load_model('model.h5')
from skimage.transform import resize
def detect(frame):
    img=resize(frame,(64,64,1))
    img=np.expand_dims(img,axis=0)
    if(np.max(img)>1):
        img = img/255.0
    predict_x=model.predict(img)
    print(predict_x)
    predict=np.argmax(predict_x,axis=1)
    x=predict[0]
    print(val[x])
frame=cv2.imread(r"C:\Users\Akshaya\PycharmProjects\Realtime_Communicati
on_System_For_Specially_Abled\Dataset\test_set\B\1.png")
data=detect(frame)
```

Flask App Building

```
import numpy as np
import os
import math
import cv2
from fer import FER
import pyttsx3
from keras.models import model_from_json
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from keras.models import load_model
from flask import Flask, render_template, Response, request
import tensorflow as tf
from cvzone.HandTrackingModule import HandDetector
from skimage.transform import resize
facecascade= cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
graph=tf.compat.v1.get_default_graph()
writer=None
model=load_model('model.h5')
font = cv2.FONT_HERSHEY_SIMPLEX
vals=['A','B','C','D','E','F','G','H','I']
emotion_detector = FER(mtcnn=True)
app=Flask(__name__,template_folder="template")
print("Accessing video stream")
app.static_folder = 'static'
vs=cv2.VideoCapture(0)
detector=HandDetector(maxHands=1)
pred=""
def SpeakText(command):
    engine = pyttsx3.init()
```



```
engine.say(command)
engine.runAndWait()
def generate_frames():
while (vs.isOpened()):
success, frame = vs.read()
hands, frame=detector.findHands(frame)
dominant_emotion, emotion_score = emotion_detector.top_emotion(frame)
if not success:
break
else:
if hands:
hand=hands[0]
x,y,w,h=hand['bbox']
imgCrop=frame[y-20:y+h+20,x-20:x+w+20]
black=np.ones((300,300,3), np.uint8)*0
ishape=imgCrop.shape
if h/w>1:
k=300/h
wcal=math.ceil(k*w)
imgresize=cv2.resize(imgCrop,(wcal,300))
irshape=imgresize.shape
wgap=math.ceil((300-wcal)/2)
black[:,wgap:wcal+wgap]=imgresize
else:
k=300/w
hcal=math.ceil(k*h)
imgresize=cv2.resize(imgCrop,(300,hcal))
irshape=imgresize.shape
hgap=math.ceil((300-hcal)/2)
```

```

black[hgap:hcal+hgap,:]=imgresize
img=resize(black,(64,64,1))
img=np.expand_dims(img,axis=0)
if(np.max(img)>1):
img = img/255.0
predict_x=model.predict(img)
classes_x=np.argmax(predict_x,axis=1)
x=classes_x[0]
SpeakText(vals[x])
dominant_emotion=str(dominant_emotion)
if(dominant_emotion!=""):
value=vals[x] + " "+ dominant_emotion
else:
value=vals[x]
cv2.putText(frame,value,(x+20,y+20),cv2.FONT_HERSHEY_SIMPLEX,
1,(255, 255, 150),2,cv2.LINE_AA)
ret, buffer = cv2.imencode('.jpg', frame)
frame = buffer.tobytes()
yield (b'--frame\r\n'
b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

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

@app.route('/sign_to_speech')
def sign_to_speech():
return render_template('sign_to_speech.html')

@app.route('/speech_to_sign')
def speech_to_sign():
return render_template('speech_to_sign.html')

```

```
@app.route('/video',methods=['GET', 'POST'])
def video():
return Response(generate_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')
if (__name__ == "__main__"):
    app.run(debug=True)
```

HTML Files

index.html

```
<!DOCTYPE
html>
<html lang="en">
<head>
<style>
body{
background-image: url("https://image.shutterstock.com/image-vector/deaf-sign-
language-welcome-line-260nw-1585429861.jpg");
background-repeat: no-repeat;
background-size: cover;
}
</style>
<script>
function loadGoogleTranslate()
{
new google.translate.TranslateElement("element")
}
</script>
<script
src="http://translate.google.com/translate_a/element.js?cb=loadGoogleTranslate"
></script>
```

```
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Document</title>
<link rel="stylesheet" href="{ { url_for('static', filename='css/index.css') } }" />
</head>
<body>
<section class="main">
<div class="inside">
<div class="wrapper">
<div class="Head">
<h1>Welcome</h1>
<span></span>
<div id="element"></div>
</div>

<a class="box1 box" href="sign_to_speech">Sign to speech</a>
<a class="box1 box" href="speech_to_sign">Speech to sign</a>
</div>
</div>
</section>
</body>
</html>
```

Speech_to_sign.html

```
<html>

<head>

<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<script
src="http://translate.google.com/translate_a/element.js?cb=loadGoogleTranslate"
></script>

<script>

function loadGoogleTranslate()

{

new google.translate.TranslateElement("google_element")

}

</script>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></scri
pt>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<style>

.row {
display: flex;
}

.col {
flex: 50%;
}

*,*:after,*:before{
-webkit-box-sizing: border-box;
-moz-box-sizing: border-box;
-ms-box-sizing: border-box;
box-sizing: border-box;
```

```
}  
body{  
font-family: arial;  
font-size: 16px;  
margin: 0;  
color: #000;  
  
display: flex;  
align-items: center;  
justify-content: center;  
min-height: 100vh;  
}  
  
.voice_to_text{  
width: 600px;  
text-align: center;  
}  
  
h1{  
color: #000000;  
font-size: 50px;  
}  
  
#convert_text{  
width: 100%;  
height: 200px;  
border-radius: 10px;  
resize: none;  
padding: 10px;  
font-size: 20px;  
margin-bottom: 10px;  
}  
  
button{
```

```
padding: 12px 20px;
background: #0ea4da;
border: 0;
color: #fff;
font-size: 18px;
cursor: pointer;
border-radius: 5px;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="container">
```

```
<div class="row">
```

```
<div class="col">
```

```

```

```
</div>
```

```
<div class="col"><div class="voice_to_text">
```

```
<div class="text_center" id="google_element"></div>
```

```
<h1>Voice to text converter</h1>
```

```
<textarea name="" id="convert_text"></textarea>
```

```
<button id="click_to_record" class="btn-primary">Voice to Text</button><br/>
```

```
<a href="/">
```

```
<button class="btn btn-danger btn-lg">Exit</button>
```

```
</a>
```

```
</div>
```

```
</div></div></div>
```

```
<script type="text/javascript" src="{ { url_for('static',
filename='javascript/script.js') } }"></script>
```

```
</body>
```

```
</html>
```

Sign_to_speech.html

```
<html>
```

```
<head>
```

```
<style>
```

```
img{
```

```
display: block;
```

```
margin-left: auto;
```

```
margin-right: auto;
```

```
}
```

```
</style>
```

```
<script>
```

```
function loadGoogleTranslate()
```

```
{
```

```
new google.translate.TranslateElement("google_element")
```

```
}
```

```
</script>
```

```
<script
```

```
src="http://translate.google.com/translate_a/element.js?cb=loadGoogleTranslate"
```

```
></script>
```

```
<script
```

```
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
```

```
<script
```

```
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></scri
```

```
pt>
```

```
<link rel="stylesheet"
```

```
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"/>
```



```
</head>

<body>
<h1>Sign to speech</h1>
<div>
<div class="text-center" id="google_element"></div>

<br/>
<div class="text-center">
<a href="/">
<button class="btn btn-danger btn-lg" >Exit</button>
</a></div>
</div>

</body>

</html>
```

CSS Files

Index.css

```
@import
url("https://fonts.googleapis.com/css2?family=Oxygen:wght@400;700&family=
Roboto:wght@300;900&display=swap");

* {
box-sizing: border-box;
padding: 0;
margin: 0;
}

:root {
--black: #000;
--white: #fff;
```

```
--hover: #000;
}

.main {
position: relative;
height: 100vh;
width: 100%;

display: flex;
align-items: center;
justify-content: center;
}

.inside {
position: relative;
height: 60%;
width: 50%;
background: rgba(255,255,255,0.9);
border-radius: 30px;
/* border: 5px solid var(--black); */
display: flex;
align-items: center;
justify-content: space-evenly;
-webkit-box-shadow: 12px 12px 17px 1px rgba(0, 0, 0, 0.59);
-moz-box-shadow: 12px 12px 17px 1px rgba(0, 0, 0, 0.59);
box-shadow: 12px 12px 17px 1px rgba(0, 0, 0, 0.59);
}

.wrapper {
position: relative;
height: 75%;
width: 30%;
```

```
display: flex;
align-items: center;
justify-content: space-evenly;
flex-direction: column;
}
```

```
.Head {
position: relative;
font-size: 3rem;
text-transform: uppercase;
font-family: "Roboto", sans-serif;
font-weight: 900;
display: flex;
align-items: center;
justify-content: center;
flex-direction: column;
height: 30%;
}
```

```
.Head h1 {
font-size: 3rem;
}
```

```
.Head span {
position: relative;
height: 5px;
width: 60%;
background: var(--black);
}
```

```
.box {
position: relative;
```

```
font-family: "Oxygen", sans-serif;
font-weight: 700;
border: 2px solid var(--black);
border-radius: 1.5rem;
text-decoration: none;
overflow: hidden;
cursor: pointer;
z-index: 1;
}
```

```
.box1 {
padding: 0.8rem 2rem;
}
```

```
.box2 {
padding: 0.8rem 1.5rem;
}
```

```
.box:hover {
color: var(--white);
background: var(--hover);
}
```

Javascript Files

Script.js

```
click_to_record.addEventListener('click',function(){
var speech = true;

const SpeechRecognition = window.speechRecognition ||
window.webkitSpeechRecognition;

const recognition =new SpeechRecognition();
recognition.interimResults = true;
recognition.addEventListener('result', e => {
```

```
const transcript = Array.from(e.results)
.map(result => result[0])
.map(result => result.transcript)
.join("")

document.getElementById("convert_text").innerHTML = transcript;
console.log(transcript);
});

if (speech == true) {
  recognition.start();
}
})
```

Output

13.2 Github and Demo Link:

<https://github.com/IBM-EPBL/IBM-Project-50400-1660906614>