A Real Time Communication System For Specially Abled

IBM NALAIYA THIRAN - PROJECT REPORT

Submitted by

M. Muddasir Ahmad - 311819104304

Mohamed Aadil. A - 311819104020

Dhanush. V - 311819104303

Sanjay Khanna. S -311819104041

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE ENGINEERING

MOHAMED SATHAK A.J. COLLEGE OF ENGINEERING

Siruseri, CHENNAI-603103

ANNA UNVIERSITY, CHENNAI-600025

NOVEMBER2022

A Real Time Communication System For Specially Abled

Team ID: **PNT2022TMID37430**

Industry Mentor(s) Name: Divya

Faculty Mentor(s) Name: Aysha Sufreen

Team Members

M. Muddasir Ahmad - 311819104304

Mohamed Aadil. A - 311819104020

Dhanush. V - 311819104303

Sanjay Khanna. S - 311819104041

Table of Contents:

1. Introduction

1.1 Project Overview

People with impairments are a part of our society. . 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 beneficial.

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.

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

● hhttp://cs231n.stanford.edu/reports/2016/pdfs/214_Report.pdf
● http://www.iosrjen.org/Papers/vol3_issue2%20(part-2)/H03224551.pdf

Other References:

●http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.8389&rep=rep1&type=pdf
● http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.8389&rep=rep1&type=pdf

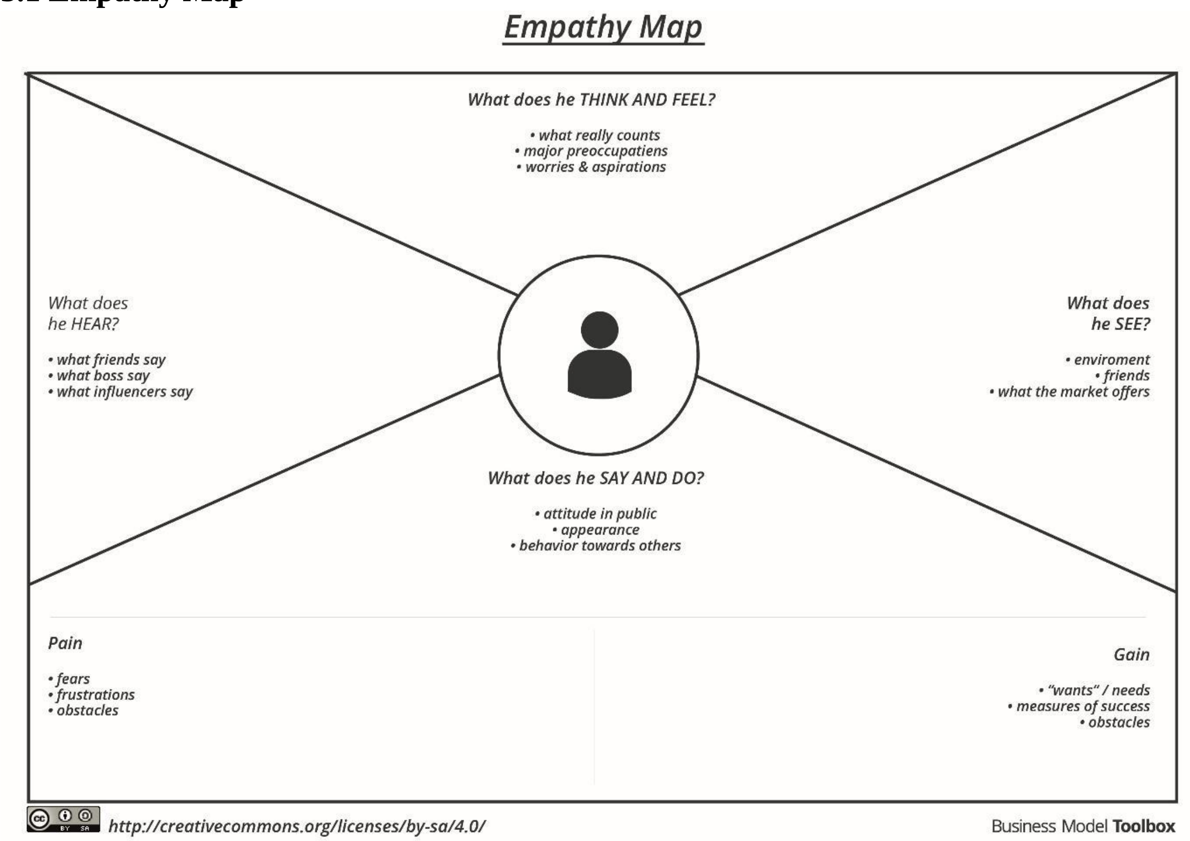
- https://ieeexplore.ieee.org/document/7507939
- https://ieeexplore.ieee.org/document/7916786

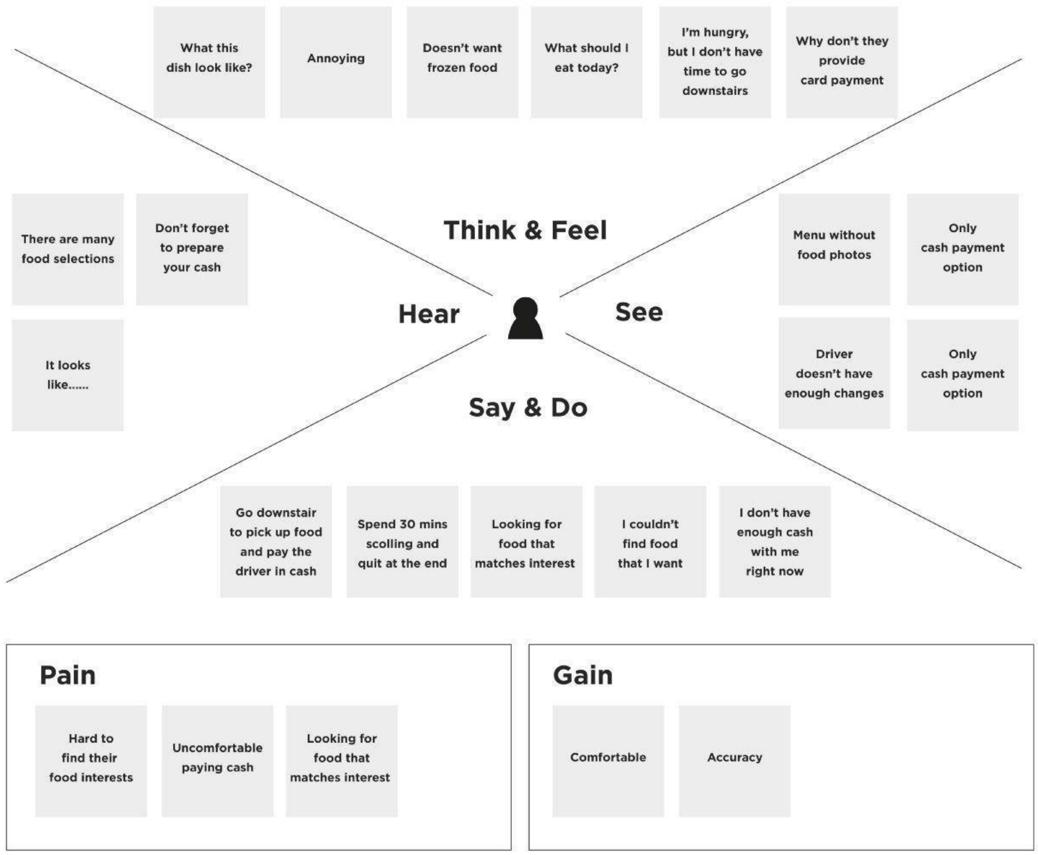
2.3 Problem Definition Statement

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 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 beneficial in establishing proper communication between a normal person and a handicapped person in any language.

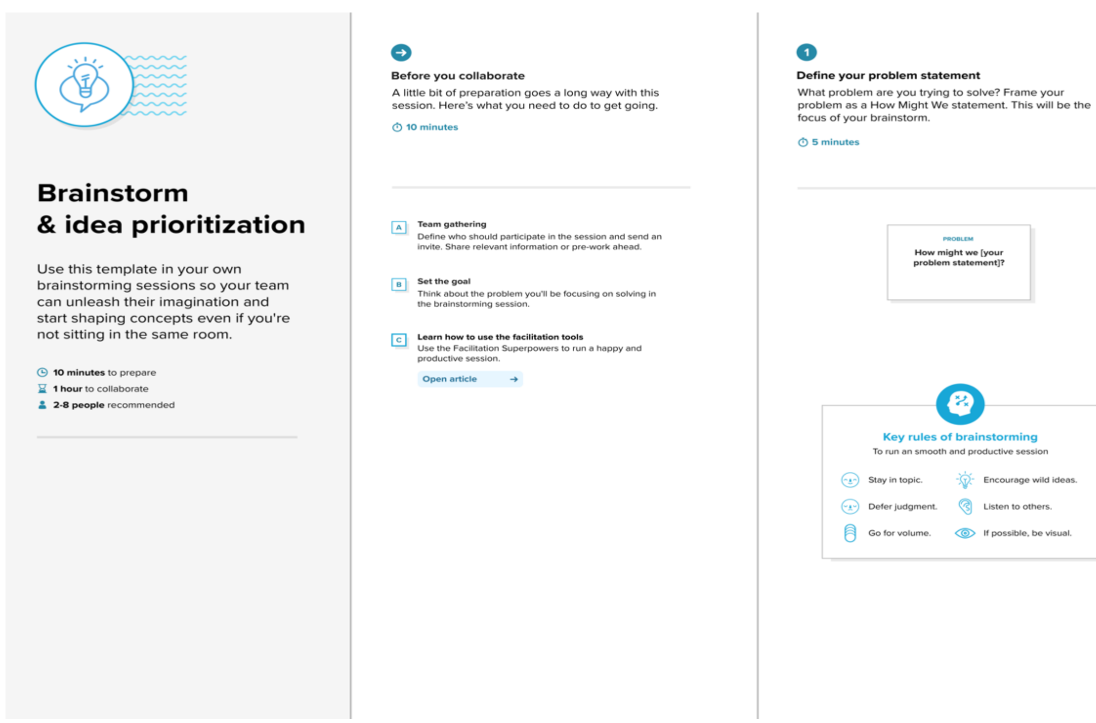
3. Ideation and Proposed Solution

3.1 Empathy Map





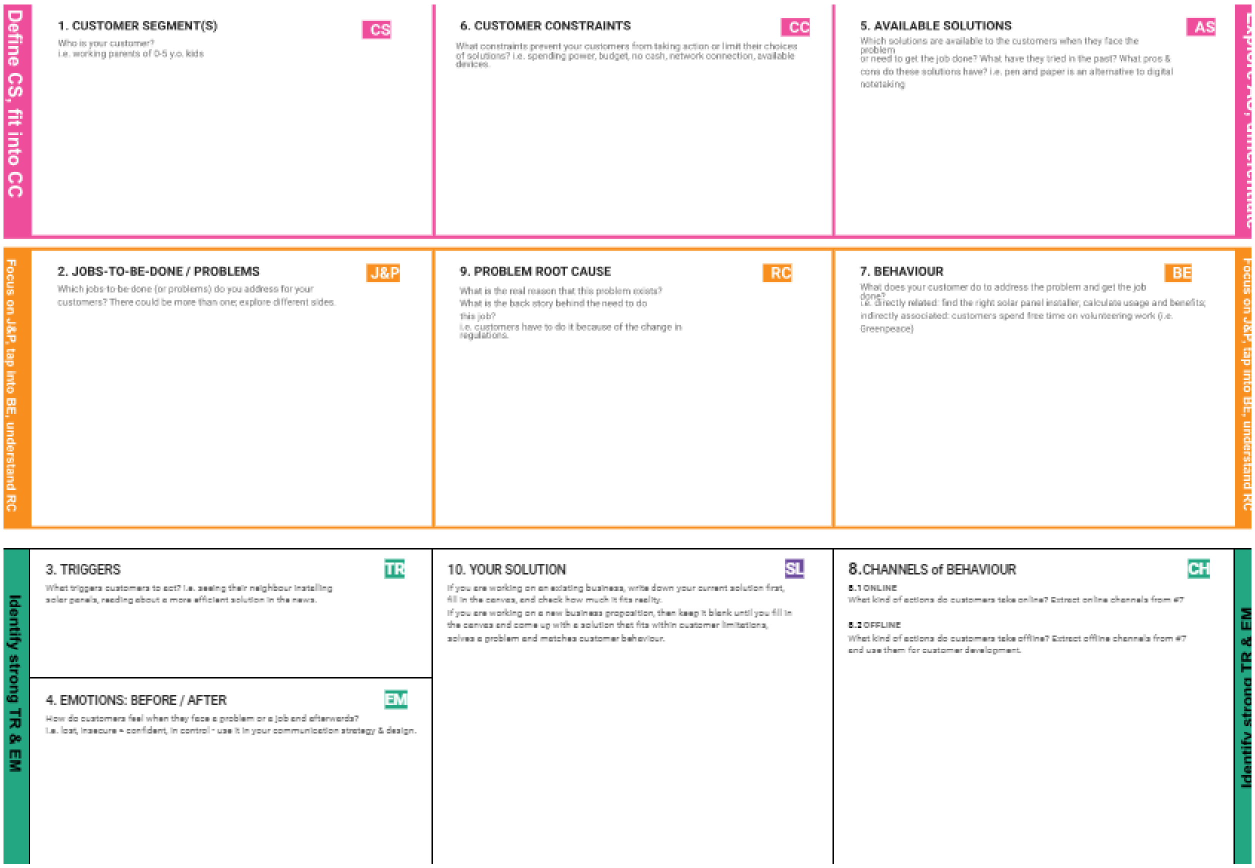
3.2 Ideation and Brainstorming



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 challenging 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.
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 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	1. Facial Emotion Detection • Language customization • User-friendly interface. 2. 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 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 extensible in terms of the features that is been offered by the system. It can be seen as a 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

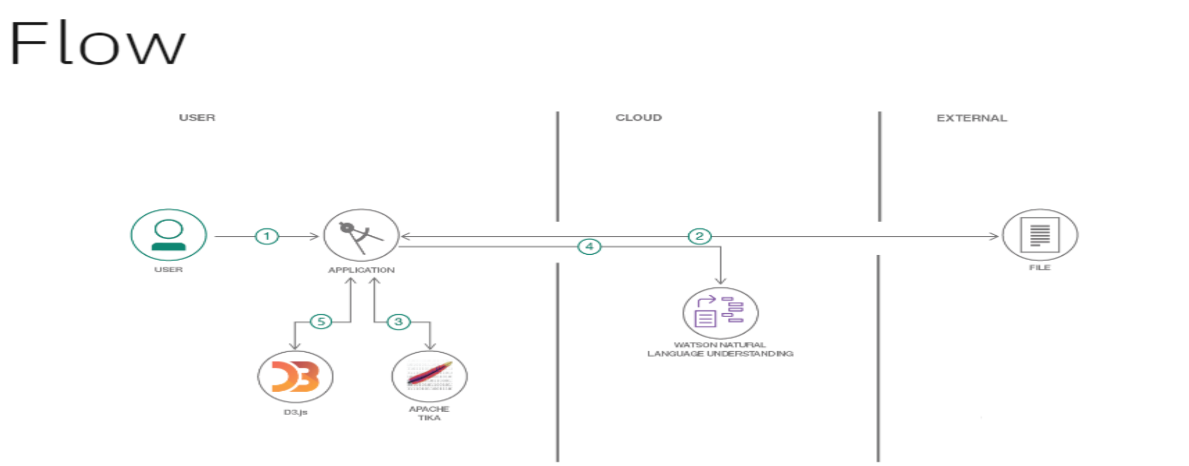
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Language customization	The user performs language customization.
FR-2	User Options	The user either chooses to convert speech to sign language and sign language to speech.
FR-3	Test Inputs	The real time video and audio data is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

4.2 Non-Functional Requirement

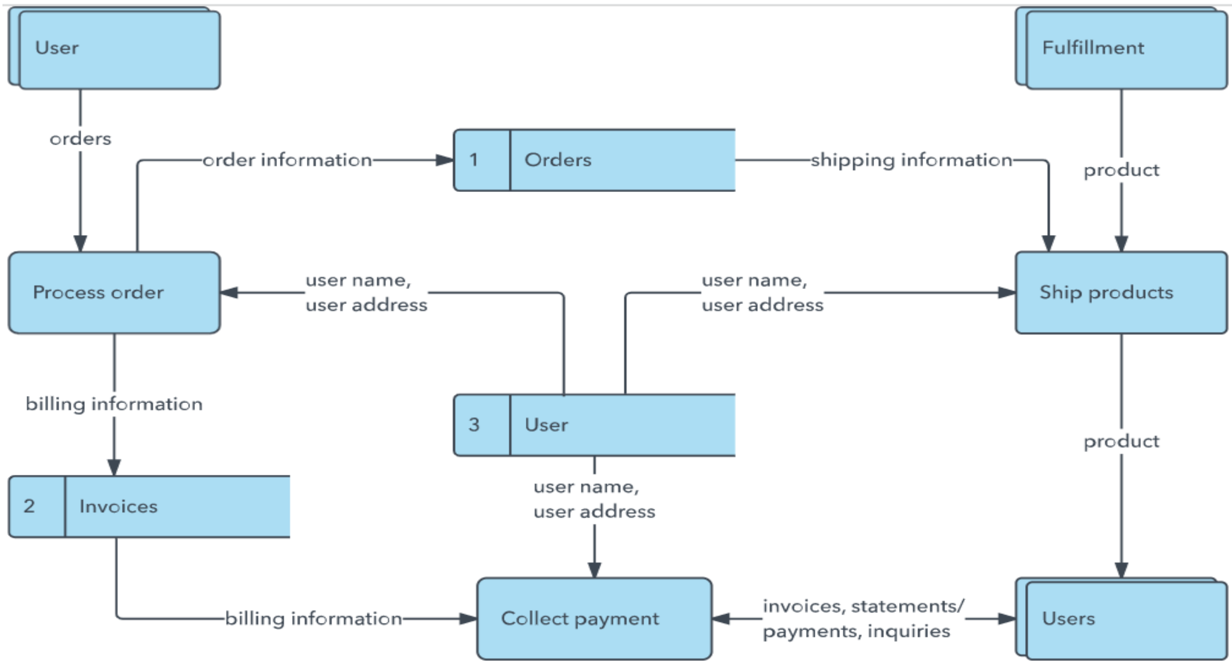
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The user will have access to all the resources present in that website.
NFR-2	Security	User information is protected.
NFR-3	Reliability	It offers accurate results.
NFR-4	Performance	The web application makes use of light weight model hence 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

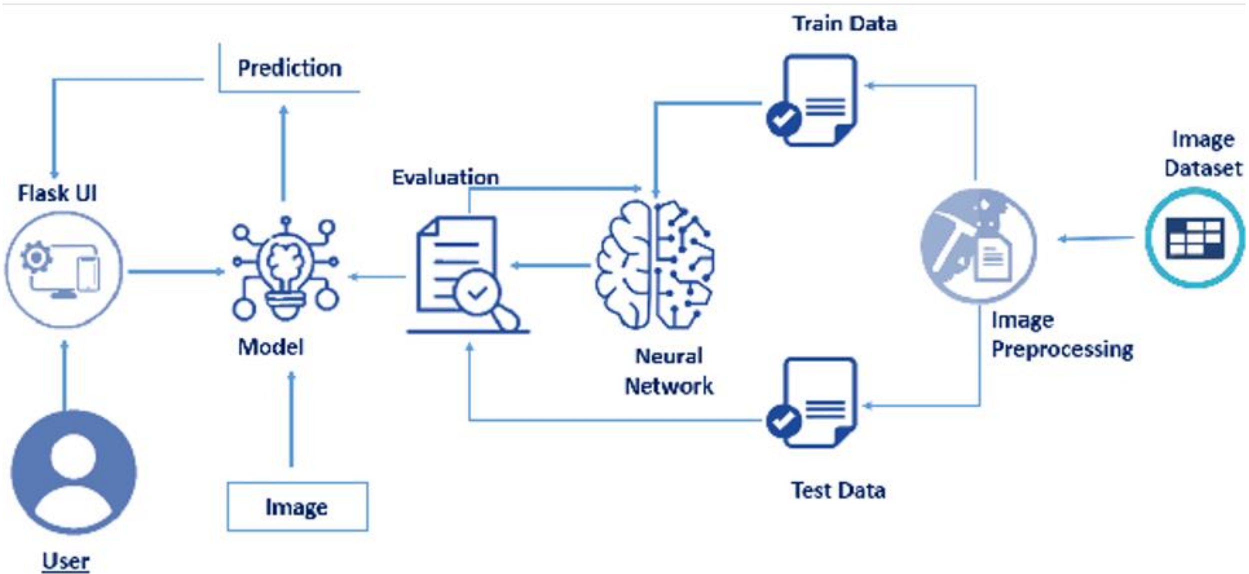
5.1 Data-flow Diagram



1. User configures credentials for the Watson Natural Language Understanding service and starts the app.
2. User selects data file to process and load.
3. Apache Tika extracts text from the data file.
4. Extracted text is passed to Watson NLU for enrichment.
5. Enriched data is visualized in the UI using the D3.js library.



5.2 Solution and Technical Architecture



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. Speech to sign language conversion. 2. Sign language to speech conversion.	They can access the portal	High	Sprint1
		USN-2	Language selection	They can access the portal	Low	Sprint1
		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	Sprint2

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 Point	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password,	2	High	Muddasir Ahmad.M Dhanush.V Sanjay Khanna.S Mohammed Aadil.A

			and confirming my password.			
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Muddasir Ahmad.M Dhanush.V Sanjay Khanna.S Mohammed Aadil.A
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	Muddasir Ahmad.M Dhanush.V Sanjay Khanna.S Mohammed Aadil.A
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	Muddasir Ahmad.M Dhanush.V Sanjay Khanna.S Mohammed Aadil.A

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burn down 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	5 Nov 2022	20	5 Nov 2022
Sprint 3	20	6 Days	7 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)

Burn down Chart:

A burn down 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.

7. Coding and Solution

7.1 Feature 1

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 complicated 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 that is capable of detection sign languages real time.

7.2 Feature 2

Real time speech to sign

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 Speech Recognition 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

1. Verify if user can see the options when user clicks the URL
2. Verify if the UI elements are getting displayed properly
3. Verify if the user can choose any languages
4. Verify if the user is getting redirected to the sign to speech page
5. Verify if the application can convert the sign to speech
6. Verify if the user can exit the sign to speech page
7. Verify if the user is getting redirected to the speech to sign page
8. Verify if the UI elements are being displayed
9. Verify if the application can convert speech to text on clicking voice to text button.
10. Verify if the user can exit the speech to sign page.

8.2 User Acceptance 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	16	13	18	71

2. Test Case Analysis

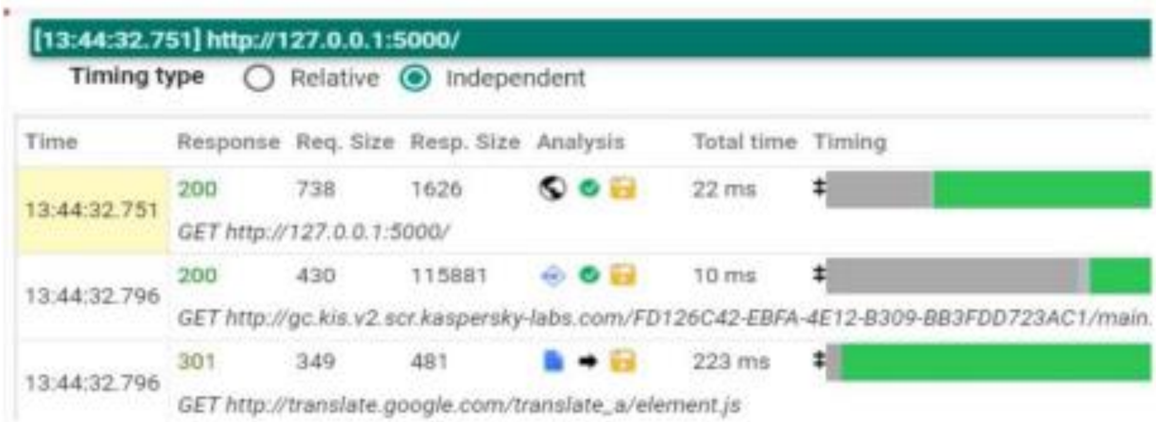
This report shows the number of test cases that have passed, failed, and untested

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

9. Results

Performance Metrics

The following images can be studied to understand the performance metrics of our system.



Time	Status	Uploaded	Downloaded	Duration	Request
13:44:33.774	200	–	–	0 ms	GET https://www.gstatic.com/images/branding/product/1x/translate_24dp.png
13:44:33.788	200	–	–	1 ms	GET https://www.gstatic.com/images/branding/googlelogo/1x/googlelogo_color_42x16dp.png
13:44:33.795	200	–	–	0 ms	GET https://www.gstatic.com/images/branding/product/2x/translate_24dp.png
13:44:33.814	200	–	–	232 ms	GET https://translate-pa.googleapis.com/v1/supportedLanguages
13:44:34.066	200	–	–	0 ms	GET https://translate.googleapis.com/translate_static/css/translateelement.css
13:44:34.067	200	–	–	0 ms	GET https://www.gstatic.com/images/branding/googlelogo/1x/googlelogo_color_68x28dp.png
13:44:34.068	200	–	–	183 ms	GET https://www.google.com/images/cleardot.gif
13:44:34.069	200	–	–	0 ms	GET https://translate.googleapis.com/translate_static/img/loading.gif
13:44:34.254	200	–	–	217 ms	GET https://www.google.com/images/cleardot.gif
13:44:34.261	200	–	–	0 ms	
13:44:34.261	200	–	–	0 ms	GET https://translate.googleapis.com/translate_static/css/translateelement.css
13:44:34.268	200	–	–	0 ms	GET https://translate.googleapis.com/translate_static/css/translateelement.css
13:44:34.295	200	–	–	0 ms	GET https://translate.googleapis.com/translate_static/img/te_bk.gif
13:44:34.296	200	–	–	0 ms	GET https://translate.googleapis.com/translate_static/img/te_ctrl3.gif
13:44:34.330	200	–	–	236 ms	POST [illegible]
13:44:34.598	204	462	1258	193 ms	GET http://translate.google.com/gen204
13:44:43.790	200	–	–	316 ms	POST https://translate.googleapis.com/element/log

Page global data

Num. requests: 28 **Uploaded:** 17346 bytes **Downloaded:** 230432 bytes **Timings:**

10. Advantages and Disadvantages

Advantages:

1. Real time sign to speech detection.

2. Model provides good accuracy.

3. Real time facial emotion detection.

4. Language Customization.

5. Real time speech to text conversion.

6. Data privacy

Disadvantages:

1. At times the website may lag.

2. Model is not tested on a wide set of data set, having all the signs.

3. Sign language customization feature is not available.

4. User cannot take notes while using the app.

5. 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 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:

1. 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.

2. The accuracy of the model shall be increased.

3. Customization of languages shall be added.

4. Users shall be allowed to write notes while on call.

5. Customization of signs can also be added as a feature.

13. Appendix Source Code

```
import cv2, pickle

import lumpy as p

import tensor-flow as if

from cnn_tf import cnn_model_fn

import is

import sqlite3

from keras.models import load_model

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

tf.logging.set_verbosity(tf.logging.ERROR)

classifier = tf.estimator.Estimator(model_dir="tmp/cnn_model2",

model_fn=cnn_model_fn)

prediction = None

model = load_model('cnn_model_keras2.h5')

def get_image_size():

I mg = cv2.imread('gestures/0/100.jpg', 0)

return img.shape

image_x, image_y = get_image_size()

def tf_process_image(img):

img = cv2.resize(img, (image_x, image_y))
```

```
img = np.array(img, d type=np.float32)
np_array = np.array(img)
return np_array
def tf_predict(classifier, image):
'''
need help with prediction using tensor flow
'''
global prediction
processed_array = tf_process_image(image)
pred_input_fn = tf.estimator.inputs.numpy_input_fn(x={"x":processed_array}, shuffle=False)
pre = classifier.predict(input_fn=pred_input_fn)
prediction = next(pred)
print(prediction)
def keras_process_image(img):
img = cv2.resize(img, (image_x, image_y))
img = np.array(img, d type=np.float32)
img = np.reshape(img, (1, image_x, image_y, 1))
return img
def keras_predict(model, image):
processed = keras_process_image(image)
pred_probab = model.predict(processed)[0]
pred_class = list(pred_probab).index(max(pred_probab))
return max(pred_probab), pred_class
def get_pred_text_from_db(pred_class):
con = sqlite3.connect("gesture_db.db")
cmd = "SELECT g_name FROM gesture WHERE
g_id="+str(pred_class)
cursor = conn.execute(cmd)
```

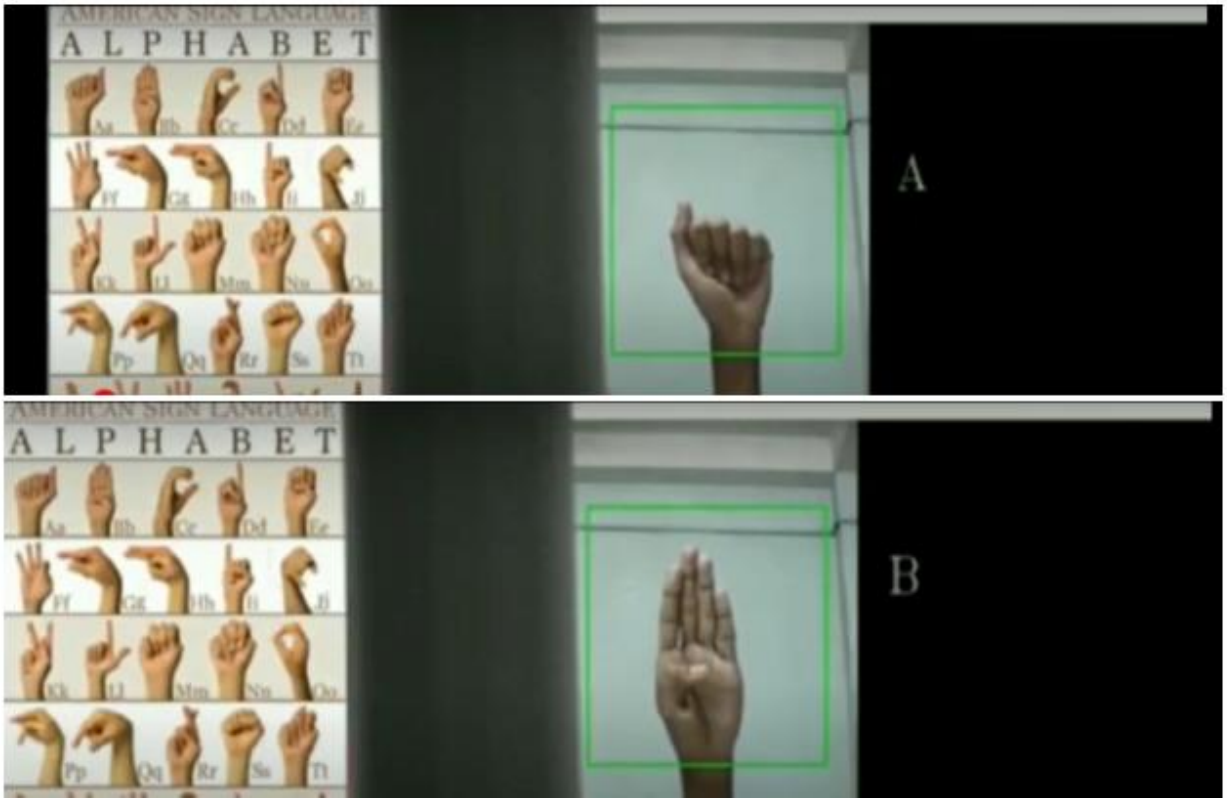
```
for row in cursor:
return row[0]
def split_sentence(text, num_of_words):
'''
Splits a text into group of num of words
'''
list_words = text.split(" ")
length = len(list_words)
splitted_sentence = []
b_index = 0
e_index = num_of_words
while length > 0:
part = ""
for word in list_words[b_index:e_index]:
part = part + " " + word
splitted_sentence.append(part)
b_index += num_of_words
e_index += num_of_words
length -= num_of_words
return splitted_sentence
def put_splitted_text_in_blackboard(blackboard, splitted_text):
y = 200
for text in splitted_text:
cv2.putText (blackboard, text, (4, y),
cv2.FONT_HERSHEY_TRIPLEX, 2, (255, 255, 255))
y += 50
def get_hand_hist():
with open("hist", "r") as f:
```

```
hist = pickle.load(f)
return hist
def recognize():
global prediction
cam = cv2.VideoCapture(1)
if cam.read()[0] == False:
cam = cv2.VideoCapture(0)
hist = get_hand_hist()
x, y, w, h = 300, 100, 300, 300
while True:
text = "" img = cam.read()[1]
img = cv2.flip(img, 1)
img = cv2.resize(img, (640, 480))
imgCrop = img[y:y+h, x:x+w]
imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
st = cv2.calcBackProject([imgHSV], [0, 1],
hist, [0, 180, 0, 256], 1)
disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(10,10))
cv2.filter2D(dst,-1,disc,st)
blur = cv2.GaussianBlur(dst, (11,11), 0)
blur = cv2.medianBlur(blur, 15)
thresh = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
thresh = cv2.merge((thresh,thresh,thresh))
thresh = cv2.cvtColor(thresh, cv2.COLOR_BGR2GRAY)
thresh = thresh[y:y+h, x:x+w]
(openCV_ver,_,__) = cv2.__version__.split(".")
if openCV_ver=='3':
contours = cv2.findContours(thresh.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)[1]
```

```
el if openCV_ver=='4':

contours = cv2.findContours(thresh.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)[0]

if len(contours) > 0:

contour = max(contours, key = cv2.contourArea)

#print(cv2.contourArea(contour))

if cv2.contourArea(contour) > 10000:

x1, y1, w1, h1 = cv2.boundingRect(contour)

save_img = thresh[y1:y1+h1, x1:x1+w1]

if w1 > h1: save_img = cv2.copyMakeBorder(save_img, int((w1-h1)/2) ,int((w1-h1)/2) , 0, 0,
cv2.BORDER_CONSTANT, (0, 0, 0)) el if h1 > w1:

save_img = cv2.copyMakeBorder(save_img, 0, 0, int((h1-w1)/2) , int((h1-w1)/2) ,
cv2.BORDER_CONSTANT, (0, 0, 0))

pred_probab,

pred_class = keras_predict(model, save_img)

if pred_probab*100 > 80:

text = get_pred_text_from_db(pred_class)

print(text) blackboard = np.zeros((480, 640, 3),

d type=np.uint8)

splitted text = split sentence (text, 2)

put_splitted_text_in_blackboard(blackboard, splitted_text)

#cv2.putText(blackboard, text, (30, 200), ‘

cv2.FONT_HERSHEY_TRIPLEX, 1.3, (255, 255, 255))

cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)

res = np.hstack((img, blackboard))

cv2.imshow("Recognizing gesture", res)

cv2.imshow("thresh", thresh)

if cv2.waitKey(1) == ord('q'):

break
```

keras_predict(model, np.zeros((50, 50), d type=np.uint8)) recognize ()

Output



13.2 GitHub and Demo Link:

https://github.com/IBM-EPBL/IBM-Project-42708-1660707358

Demo Link:

https://drive.google.com/file/d/1Yw1uTOi5YborPnG8Y4joLJlC_VD2Sjif/view?usp=drivesdk