PROJECT REPORT
REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED

TEAM ID: PNT2022TMID14822

TEAM MEMBERS: AKILANDESWARI.S – 111519106004
ASWITHA.K.G – 111519106007
BHAVYA.L – 111519106013
DEEPIKA.T – 111519106022

1.INTRODCUTION:

1.1 PROJECT OVERVIEW:

Real-time communications (RTC) is any mode of telecommunications in which all users can exchange information instantly or with negligible latency or transmission delays. In RTC, there is always a direct path between the source and the destination. Although the link might contain several intermediate nodes, the data goes from source to destination without being stored in between them. In contrast, asynchronous or time shifting communications, such as email and voicemail, always involve some form of data storage between the source and the destination. In these cases, there is an anticipated delay between the transmission and receipt of the information.

1.2 PROBLEM STATEMENT:

The Deaf and mute community can only communicate using sign language. Sign language involves simultaneously combining hand shapes, orientations, gestures and movement of the hands, arms, or body to express the speaker's thoughts. Because of cultural, geographic and historical differences, there exists over 300 different types of sign languages around the world. The ISL (Indian Sign Language) used in India is very different from the American Sign Language used in the United States. This causes inconsistency of sign languages around the world. Moreover, learning sign language requires significant amount of time and effort. This makes it difficult for the conventional world to learn and hence interact with the deaf and mute community. According to a recent study, out of every thousand kids born, 2 to 3 of them are deaf or hard-of-hearing, and, as degrees of hearing loss go, there are 16 to 30 times more children who are identified as Deaf (having a Profound 91+dB hearing loss) than hard-of-hearing. For those deaf or hard of hearing children, only 10% of parents & family learn sign language to communicate with them. We identify this as a major barrier in communicating with a significant part of the society.

2. LITERATURE SURVEY:

2.1 PROBLEM STATEMENT DEFINITION:

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. 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 isgivenasoutput.

2.2 REFERENCES:

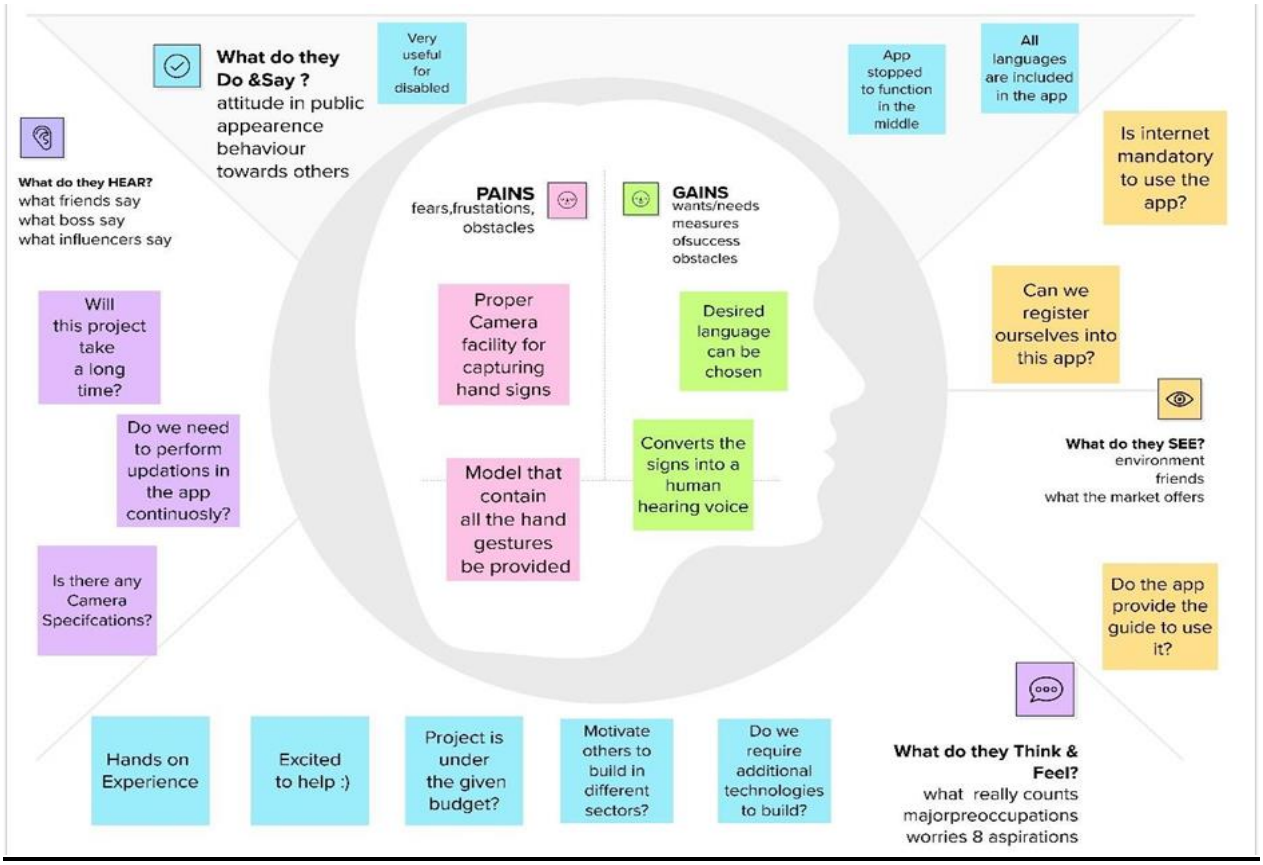
S. No	TITLE	AUTHOR	JOURNAL	TECHNIQUES	FINDINGS	YEAR
1	CIRCA: A cooperative intelligent realtime control architecture	Musliner, David J and Durfee, Edmund H and Shin, Kang G	IEEE Transactions on Systems, Man, and Cybernetics	The Cooperative Intelligent Realtime Control Architecture (CIRCA)	We have applied a prototype CIRCA implementation to a simulated Puma robot arm performing multiple tasks with real-time deadlines	1993

2	The challenges of real-time AI	Musliner, David J and Hendler, James A and Agrawala, Ashok K and Durfee, Edmund H and Strosnider, Jay K and Paul, CJ	Computer	Embedding AI in real time	Found that the broad application of AI methods to real-time domains will require new approaches, diering from many of the traditional search-based techniques explored in the field.	1995
3	High-speed railway communications: From GSM-R to LTE-R	He, Ruisi and Ai, Bo and Wang, Gongpu and Guan, Ke and Zhong, Zhangdui and Molisch, Andreas F and BrisoRodriguez, Cesar and Oestges, Claude P	Ieee vehIcular technology magazIne	GSM-R, LTE, and LTE-R	Provides an overview of HSR-dedicated communication systems	2016
4	Real-time scheduling for energy harvesting sensor nodes	Moser, Clemens and Brunelli, Davide and Thiele, Lothar and Benini, Luca	Real-Time Systems	LSA-I algorithm,LSA-II algorithm	The arrival times, energy demands and deadlines	2007

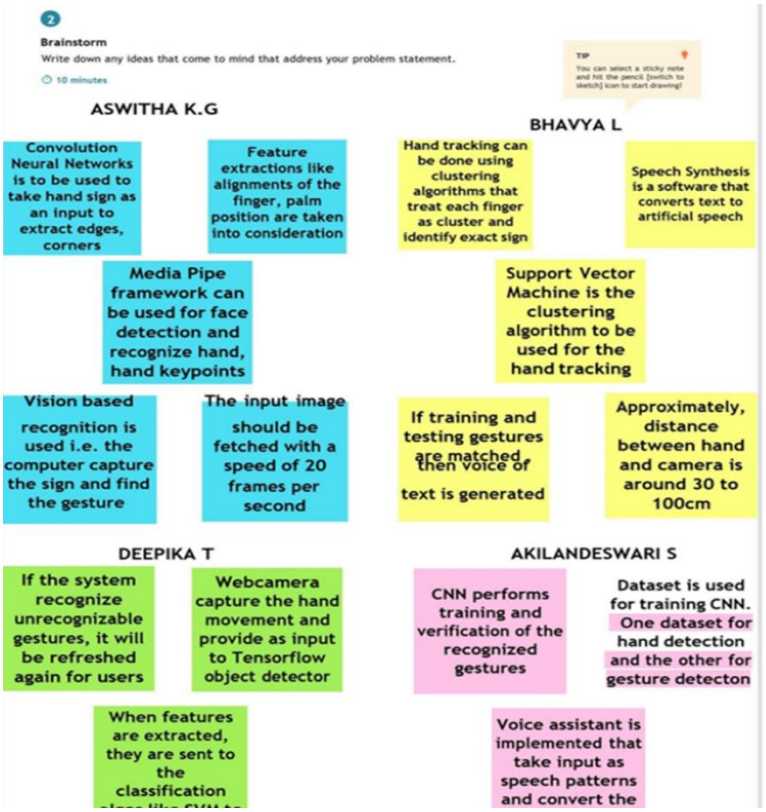
5	Designing the next generation of realtime control, communication, and computations for large power systems	Tomsovic, Kevin and Bakken, David E and Venkatasub ramanian, Vaithianath an and Bose, Anjan	Proceedings of the IEEE	Decentralized Load Frequency Control with AGC	To control the dynamics directly without having to set special protection parameters	2005
6	Real-time knowledge-based systems	Laffey, Thomas J and Cox, Preston A and Schmidt, James L and Kao, Simon M and Readk, Jackson Y	AI magazine	The Hybrid Expert System Controller (Hexscon),Fuzzy Inference Chip	Real-time problem solving, many human limitation	1988

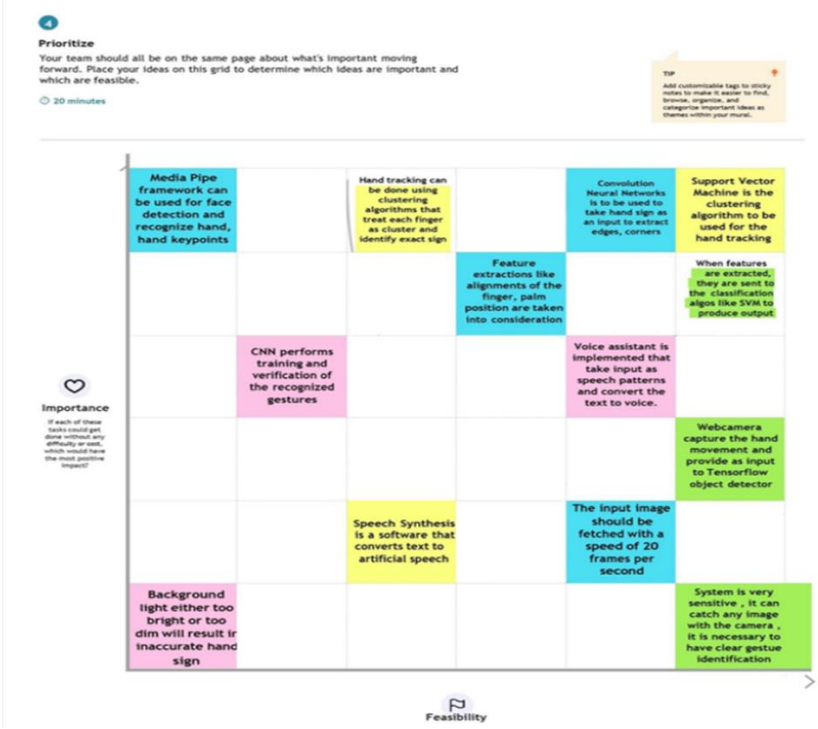
3. IDEATION AND PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:



3.2 IDEATION & BRAINSTORMING:

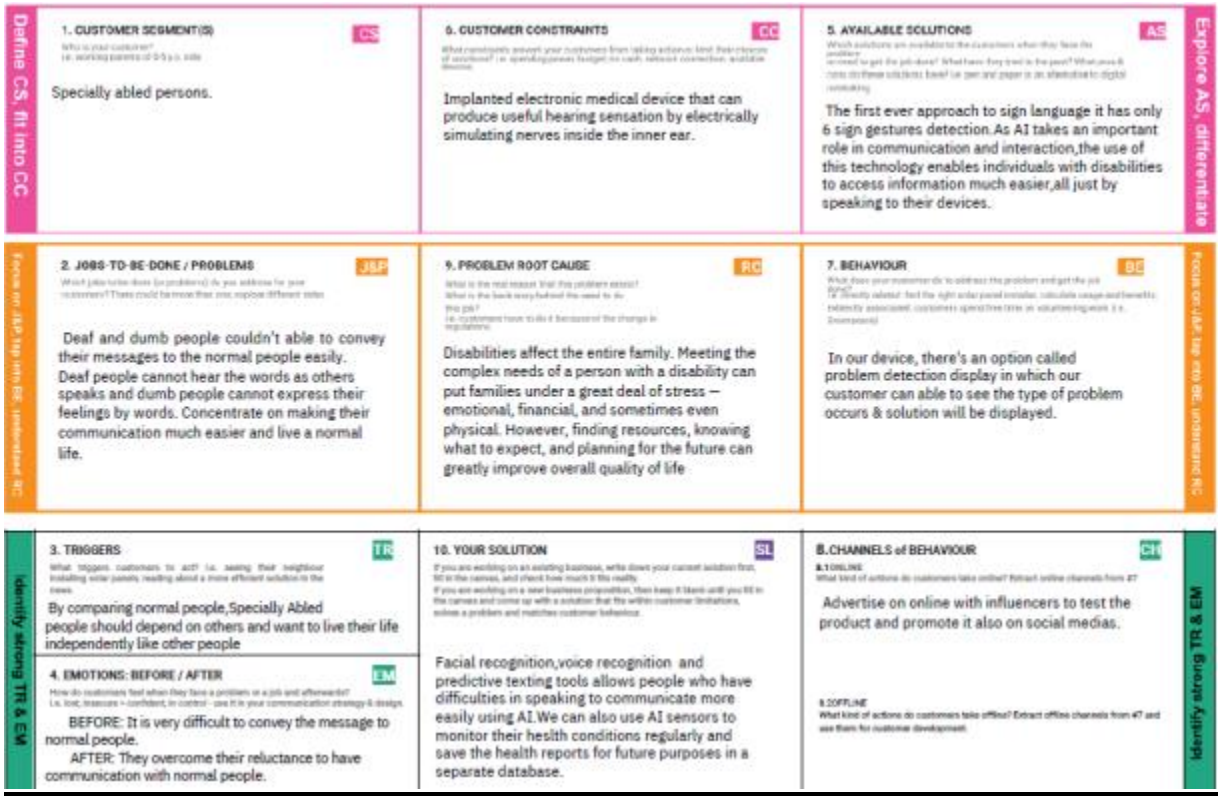




3.3. PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Differently able like dump and mute people can communicate only through the sign language ,normal people those who do not know the sign language feels difficult to communicate with them.
2.	Idea / Solution description	To overcome this problem we have an idea that an application is created to communicate with the normal people.
3.	Novelty / Uniqueness	This process the image of the person who is using sign language and convert it into the voice by analyzing the sign used.
4.	Social Impact / Customer Satisfaction	Differently able people feel free to communicate and it bring a huge difference comparing to past.
5.	Business Model (Revenue Model)	There are many people in the world who is differently able,this application will become more popular among them and it will be installed by all and it will be used,and so it will produce more money.
6.	Scalability of the Solution	Thus this would bring a new evolution in Real Time Communication System Powered by AI for Specially Able with less time and safe enough resources

3.4 PROBLEM SOLUTION FIT:



4. REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENTS:

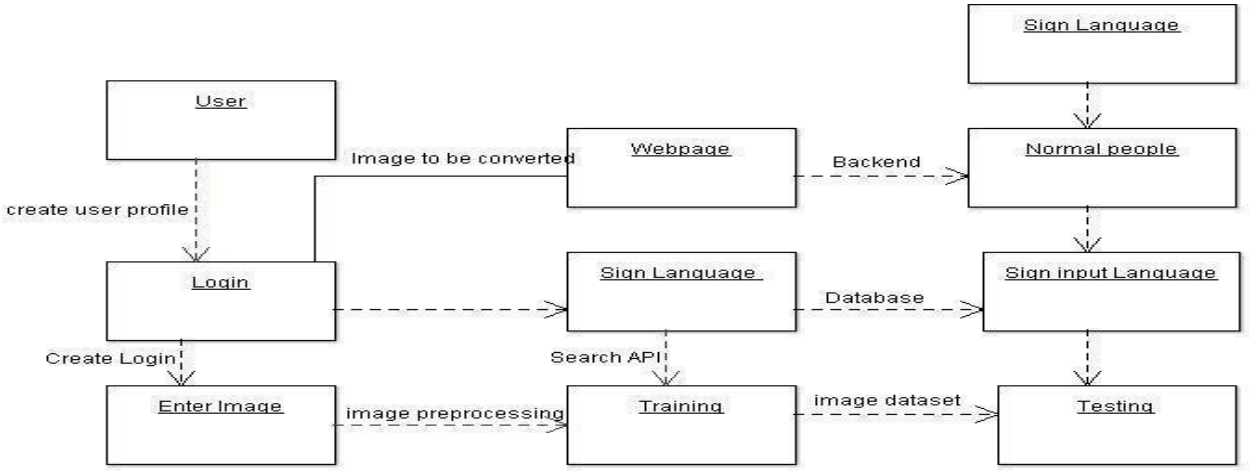
	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	**LOW VISION:** As a user who has trouble reading due to low vision, I want to be able to make the text larger on the screen so that I can read it. **Registration through Gmail**
FR-2	User Confirmation	**IMPAIRED USER:** As a user who is hearing -impaired, I want a turn on video captions so that I can understand what is being said in videos. **Confirmation via Email**
FR-3	User Registration	**COLOR BLINDNESS**: As a user who is color blind, I want to links to be distinguishable on the page so that I can find the links and navigate the site. **Registration through Gmail**

4.2 NON FUNCTIONAL REQUIREMENTS:

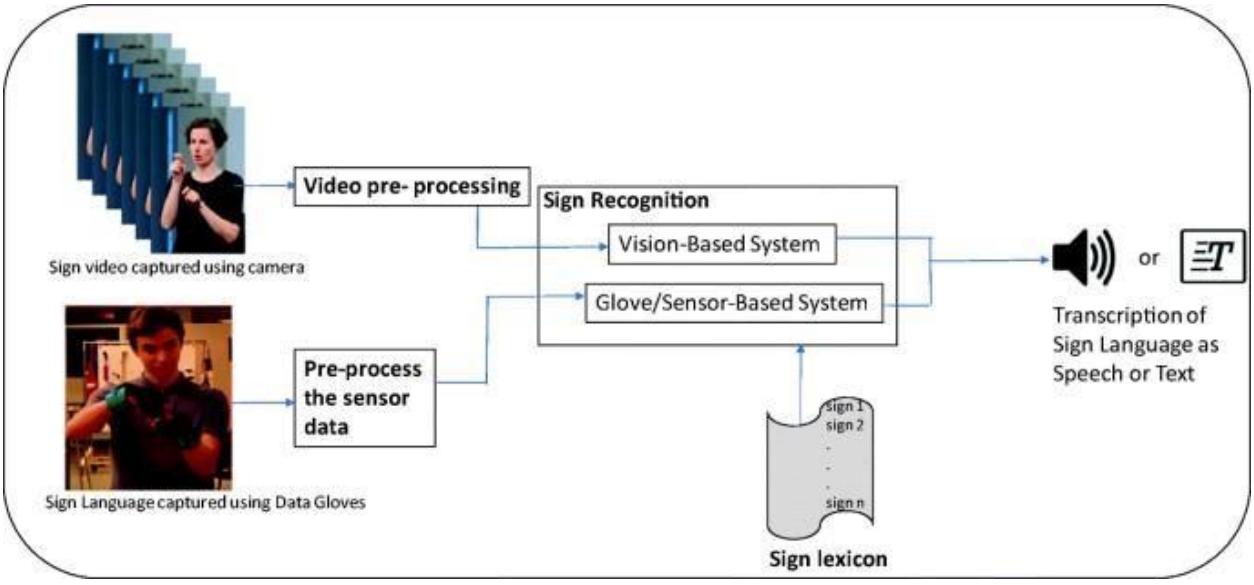
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• Visual and Audio Help • Text size scaling • Reverse contrast
NFR-2	**Security**	Important information: • Walking in single file or in narrow space. • Steps, Stairs and Slope. • Kerbs and Roads.
NFR-3	**Reliability**	To determine reliability measures are: • Test-Retest Repeatability • Individual Repeatability
NFR-4	**Performance**	To determine predictors of success in reading with low vision aids, in terms of reading acuity, optimum acuity reserve, and maximum reading speed, for observers with low vision for various causes.
NFR-5	**Availability**	Lack of adequate low vision services and barriers to their provision and uptake impact negatively on efforts to prevent visual impairment and blindness.
NFR-6	**Scalability**	There is a large selection of device to help people with low vision. Some are “Optical”, glass lenses such as magnifying glasses and telescopes.

5.PROJECT DESIGN:

5.1 DATA FLOW DIAGRAMS:



5.2 SOLUTION AND TECHNOLOGY ARCHITECTURE:



5.3 USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Low vision)	Registration	USN-1	As a user, who has trouble reading due to low vision, I want to be able to make the text larger on the screen so that I can read it.	I can access my account / dashboard	High	Sprint-1
Customer (Color blindness)		USN-2	As a user, who is color blind ,I want to have access to information conveyed in color so that, I do not miss anything and I understand the content.	I can receive confirmation email & click confirm	High	Sprint-1
Customer (Impaired user)		USN-3	As a user, who is hearing - impaired, want a transcript of the spoken audio so that I can have access to all information provided in audio clips.	I can register & access the dashboard with Facebook Login	Low	Sprint-2

6. PROJECT PLANNING & SCHEDULING:

6.1 SPRINT DELIVERY PLAN:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Data Collection	USN-1	Collect Dataset .	9	High
Sprint-1	Image processing	USN-2	Image preprocessing	8	Medium
Sprint-2	Model Building	USN-3	Import the required libraries, add the necessary layers and compile the model	10	High
Sprint-2		USN-4	Training the image classification model using CNN	7	Medium
Sprint-3	Training and Testing	USN-5	Training the model and testing the model's performance	9	High
Sprint-4	Implementation of the application	USN-6	Converting the input sign language images into English alphabets	8	Medium

7.CODING AND SOLUTIONING:

MODEL BUILDING:

IMPORT MODELLIBRARY:

In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
from keras.utils import np_utils
```

In [3]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [4]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'./training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'./test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15130 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [5]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  17
Len x-test :  3
```

In [6]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[6]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [7]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
```

INITIALIZE THE MODEL:

In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
spatial_dropout=0.05
recurrent_dropout=0.1
```

In [3]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [4]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'./training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'./test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15130 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [5]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  17
Len x-test :  3
```

In [6]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[6]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [7]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [8]:
```
# Creating Model
model=Sequential()
```

ADD THE CONVOLUTION LAYERS:

In [1]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [2]:
```
x_train = train_datagen.flow_from_directory('./training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 15130 images belonging to 9 classes.
```

In [3]:
```
x_test = test_datagen.flow_from_directory('./test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 2250 images belonging to 9 classes.
```

In [4]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
```

In [5]:
```
model = Sequential()
```

In [6]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1), activation='relu'))
#no. of feature detectors, size of feature detector, image size, activation function
```

ADD THE POOLING LAYER:

In [1]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [2]:
```
x_train = train_datagen.flow_from_directory('./training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 15130 images belonging to 9 classes.
```

In [3]:
```
x_test = test_datagen.flow_from_directory('./test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 2250 images belonging to 9 classes.
```

In [4]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
```

In [5]:
```
model = Sequential()
```

In [6]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1), activation='relu'))
#no. of feature detectors, size of feature detector, image size, activation function
```

In [7]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

ADD THE FLATTEN LAYERS:

In [1]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [2]:
```
x_train = train_datagen.flow_from_directory('./training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 15130 images belonging to 9 classes.
```

In [3]:
```
x_test = test_datagen.flow_from_directory('./test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 2250 images belonging to 9 classes.
```

In [4]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
```

In [5]:
```
model = Sequential()
```

In [6]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1), activation='relu'))
#no. of feature detectors, size of feature detector, image size, activation function
```

In [7]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [8]:
```
model.add(Flatten())
```

ADD THE DENSE LAYER:

In [1]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [2]:
```
x_train = train_datagen.flow_from_directory('./training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 15130 images belonging to 9 classes.
```

In [3]:
```
x_test = test_datagen.flow_from_directory('./test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 2250 images belonging to 9 classes.
```

In [4]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
```

In [5]:
```
model = Sequential()
```

In [6]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1), activation='relu'))
#no. of feature detectors, size of feature detector, image size, activation function
```

In [7]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [8]:
```
model.add(Flatten())
```

In [9]:
```
model.add(Dense(units=512, activation = 'relu'))
```

In [10]:
```
model.add(Dense(units=9,  activation = 'softmax'))
```

COMPILE THE MODEL:

In [1]:
```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [2]:
```
x_train = train_datagen.flow_from_directory('./training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 15130 images belonging to 9 classes.
```

In [3]:
```
x_test = test_datagen.flow_from_directory('./test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
```

```
Found 2250 images belonging to 9 classes.
```

In [4]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
```

In [5]:
```
model = Sequential()
```

In [6]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1), activation='relu'))
#no. of feature detectors, size of feature detector, image size, activation function
```

In [7]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [8]:
```
model.add(Flatten())
```

In [9]:
```
model.add(Dense(units=512, activation = 'relu'))
```

In [10]:
```
model.add(Dense(units=9,  activation = 'softmax'))
```

In [11]:
```
model.compile(loss='categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

FIT & SAVE THE MODEL:

In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [3]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'./training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'./test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15130 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [4]:
```
# Save Model Using Pickle
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
import pickle
```

In [5]:
```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size,
random_state=seed)
```

In [6]:
```
# Fit the model on training set
model = LogisticRegression(max_iter=10000)
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))

# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, Y_test)
print(result)
```

```
0.7874015748031497
```

In [7]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  17
Len x-test :  3
```

In [8]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[8]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [9]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [10]:
```
# Creating Model
model=Sequential()
```

In [11]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [12]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [13]:
```
model.add(Flatten())
```

In [14]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [15]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [16]:
```
# Fitting the Model Generator
model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
WARNING:tensorflow:From :2: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
17/17 [==============================] - 77s 5s/step - loss: 1.0774 - accuracy: 0.6352 - val_loss: 0.3910 - val_accuracy: 0.9102
Epoch 2/10
17/17 [==============================] - 73s 4s/step - loss: 0.2201 - accuracy: 0.9365 - val_loss: 0.2462 - val_accuracy: 0.9449
Epoch 3/10
17/17 [==============================] - 73s 4s/step - loss: 0.0951 - accuracy: 0.9746 - val_loss: 0.1913 - val_accuracy: 0.9693
Epoch 4/10
17/17 [==============================] - 72s 4s/step - loss: 0.0501 - accuracy: 0.9874 - val_loss: 0.1801 - val_accuracy: 0.9729
Epoch 5/10
17/17 [==============================] - 71s 4s/step - loss: 0.0294 - accuracy: 0.9927 - val_loss: 0.1693 - val_accuracy: 0.9778
Epoch 6/10
17/17 [==============================] - 74s 4s/step - loss: 0.0199 - accuracy: 0.9958 - val_loss: 0.2162 - val_accuracy: 0.9756
Epoch 7/10
17/17 [==============================] - 72s 4s/step - loss: 0.0128 - accuracy: 0.9973 - val_loss: 0.1827 - val_accuracy: 0.9787
Epoch 8/10
17/17 [==============================] - 72s 4s/step - loss: 0.0092 - accuracy: 0.9985 - val_loss: 0.2153 - val_accuracy: 0.9787
Epoch 9/10
17/17 [==============================] - 74s 4s/step - loss: 0.0073 - accuracy: 0.9987 - val_loss: 0.1976 - val_accuracy: 0.9787
Epoch 10/10
17/17 [==============================] - 71s 4s/step - loss: 0.0058 - accuracy: 0.9989 - val_loss: 0.2034 - val_accuracy: 0.9782
```

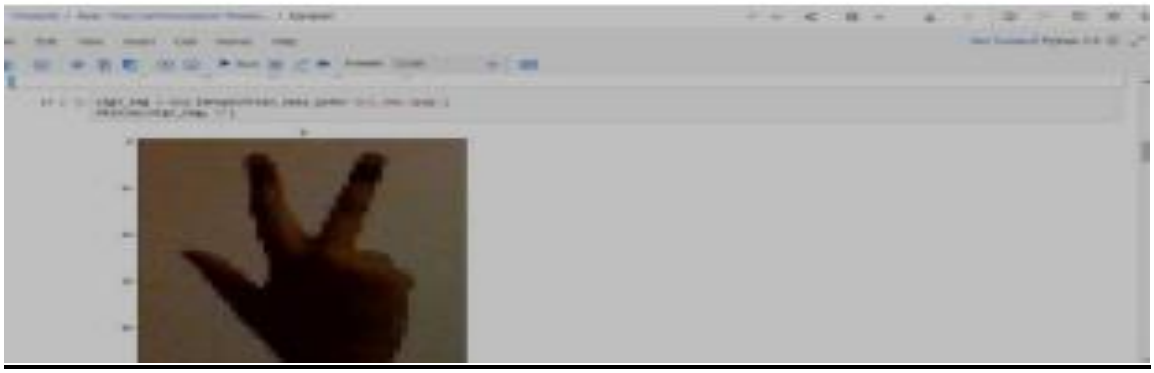
Out[16]:

Saving the Model

In [17]:
```
model.save('asl_model_84_54.h5')
```

9.RESULTS:

PERFORMANCE METRICS:



10.ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

• Segmentation accuracy is high

• Easy to detect the finger postures

• Track fingers and sign recognition with less computational steps

• No need for additional hardware system

DISADVANTAGES:

• Need hardware control to detect the hands

• Hand segmentation become complex of various backgrounds

• Segmentation accuracy is less in hand tracking

11. CONCLUSION:

The ability to look, listen, talk, and respond appropriately to events is one of the most valuable gifts a human being can have. However, some unfortunate people are denied this opportunity. People get to know one another through sharing their ideas, thoughts, and experiences with others around them. There are several ways to accomplish this, the best of which is the gift of "Speech." Everyone can very persuasively transfer their thoughts and comprehend each other through speech. Our initiative intends to close the gap by including a low-cost computer into the communication chain, allowing sign language to be captured, recognized, and translated into speech for the benefit of blind individuals. An image processing technique is employed in this paper to recognize the handmade movements. This application is used to present a modern integrated planned system for hear impaired people. The camera-based zone of interest can aid in the user's data collection. Each action will be significant in its own right.

12. FUTURE SCOPE:

Despite it having average accuracy, our system is still well-matched with the existing systems, given that it can perform recognition at the given accuracy with larger vocabularies and without an aid such as gloves or hand markings. In future, we can extend the framework to implement various deep learning algorithms to recognize the signs and implement in real time applications.

13. APPENDIX:

SOURCE CODE:

```
import tensorflow as tf
import numpy as np
import trainlist
import cv2


model=tf.keras.models.load_model("./Model/sign_1.h5")
image=tf.keras.preprocessing.image


#print(model.summary())


fl_img='./Data/Test/G/Image_1667714982.6115465.jpg'
img=image.load_img(fl_img,target_size=(224,224))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=np.argmax(model.predict(x))
op=trainlist.dataset
ans=op[pred]
print("\n\t"+ans+"\n")


import cv2 as cv
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
```

```
from cvzone.ClassificationModule import Classifier
import trainlist

cap=cv.VideoCapture(0)
detector = HandDetector(maxHands=1)
offset=20
img_size=300
classifier=Classifier("./Model/sign_1.h5","./Model/labels.txt")
labels=trainlist.dataset
list=[" "]
count=0

while True:
    ret,img=cap.read()
    img_out=img.copy()
    hands,img=detector.findHands(img)
    if hands:
        hand=hands[0]
        x,y,w,h=hand['bbox']
        #Image empty
        img_bg=np.ones((img_size,img_size,3), np.uint8)*255
        croped_img=img[y-offset:y+ h+offset,x-offset:x+ w+offset]

        aspect_ratio=h/w
```

```
        if aspect_ratio>1:
            k=img_size/h
            wCal= math.ceil(k*w)
            img_resize=cv.resize(croped_img,(wCal,img_size))
            wGap =math.ceil((img_size-wCal)/2)
            img_bg[:,wGap:wCal+wGap] = img_resize
            prediction,index=classifier.getPrediction(img_bg)
            print(labels[index])

        else:
            k=img_size/w
            hCal= math.ceil(k*h)
            img_resize=cv.resize(croped_img,(img_size,hCal))
            hGap =math.ceil((img_size-hCal)/2)
            img_bg[hGap:hCal+hGap,:] = img_resize
            prediction,index=classifier.getPrediction(img_bg)
            print(labels[index])

        cv.putText(img_out,labels[index],(x,y-
20),cv.FONT_HERSHEY_COMPLEX,2,(255,255,255),2)

        cv.imshow("Image_croped",croped_img)
        cv.imshow("Image_bg",img_bg)

        gesture=labels[index]
        count+=1
```

```
        if count==30:
            if gesture!=list[-1]:
                list.append(gesture)
            count=count-30
        print(list)
    cv.imshow("Image",img_out)
    key=cv.waitKey(1)

    if key==ord('q'):
        break

cap.release()
cv.destroyAllWindows()

<<<<<<< HEAD
from flask import Flask, Response, render_template
import cv2



app = Flask(__name__)
cap = cv2.VideoCapture(0)
@app.route('/')
def index():
    return  render_template('index.html')
```

```
def generate_frames():
    while True:
        success, frame = cap.read()
        imgOutput=frame.copy()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + imgOutput + b'\r\n')
@app.route('/predict',methods=['POST','GET'])
def predictions():
    #The prediction model code goes here
    #Once the start Button is pressed the prediction model starts
    pass
@app.route('/stop',methods=['POST','GET'])
def stopping():
    #The text to speech code goes here
    #Once the stop button is pressed the text is converted into speech
    pass


@app.route('/view.py')
def video():
    return Response(generate_frames(),mimetype='multipart/x-mixed-replace;
boundary=frame')




if __name__ == '__main__':
=======
from flask import Flask, Response, render_template
```

```
import cv2



app = Flask(__name__)
cap = cv2.VideoCapture(0)
@app.route('/')
def index():
    return  render_template('index.html')


def generate_frames():
    while True:
        success, frame = cap.read()
        imgOutput=frame.copy()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + imgOutput + b'\r\n')
@app.route('/predict',methods=['POST','GET'])
def predictions():
    #The prediction model code goes here
    #Once the start Button is pressed the prediction model starts
    pass
@app.route('/stop',methods=['POST','GET'])
def stopping():
    #The text to speech code goes here
    #Once the stop button is pressed the text is converted into speech
    pass
```

```
@app.route('/view.py')

def video():

    return Response(generate_frames(),mimetype='multipart/x-mixed-replace;
boundary=frame')




if __name__ == '__main__':

>>>>>>> 61026829fa57792e43947b1534dee2443e6f9852

        app.run(debug=True)
```

```
<!doctype html>

<html lang="en">

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-
fit=no">

    <title>VHearuU</title>

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">

    <link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">

    <link rel="stylesheet" href="static/css/Banner-Heading-Image.css">

    <link rel="stylesheet" href="static/css/Navbar-Centered-Brand.css">

    <link rel="stylesheet" href="static/css/styles.css">

</head>
```

```
<body>


<nav class="navbar navbar-light navbar-expand-md py-3" style="background:
#429691;">

    <div class="container">

        <div></div><a class="navbar-brand d-flex align-items-center" href="#"><h4
style="color: #ffffff; font-style: oblique; text-align: center;font-family: Arial
Black"><strong> Real-Time Communication

            System Powered By AI&amp;nbsp;For Specially Abled - Team ID:
PNT2022TMID47801</strong></h4></a>

        <div></div>

    </div>

  </nav>

  <section>

    <div class="d-flex flex-column justify-content-center align-items-center">

        <div class="d-flex flex-column justify-content-center align-items-center" id="div-
video-feed"

            style="width: 800px;height: 600px;margin: 10px;min-height: 480px;min-width:
640px;border-radius: 10px;border: 5px groove #000000 ;">

            <img src="{{ url_for('video_feed') }}" style="width: 100%;height: 100%;color:
rgb(255,255,255);text-align: center;font-size: 20px;"

                alt="Camera Access Not Provided!">

        </div>

    </div>

  </section>

<section>

    <div class="container">

        <div class="accordion text-white" role="tablist" id="accordion-1">
```

```
            <div class="accordion-item" style="font-style: oblique; background:
#429691;">

                <h2 class="accordion-header" role="tab"><button class="accordion-button
collapsed"

                    data-bs-toggle="collapse" data-bs-target="#accordion-1 .item-2" aria-
expanded="false"

                    aria-controls="accordion-1 .item-2"

                    style="font-style: oblique; background: #cc7931;color: #ffffff;">Project:
Real-Time Communication System Powered by AI for Specially Abled</button></h2>

                <div class="accordion-collapse collapse item-2" role="tabpanel" data-bs-
parent="#accordion-1">

                    <div class="accordion-body">

                        <p class="mb-0">Team ID: PNT2022TMID47801<br><br>VHearU
Developed By,<br><br>1. <strong>AbdulMubarak R</strong> 911719104701<br>2.

                            <strong>Mohanasrinath K</strong> 911719104035<br>3.
<strong>Abdul Azeez N</strong> 911719104001<br>4. <strong>karuppaiah S</strong>
911719104024<br>5. <strong> Pragadheesh K</strong> 911719104046

                        </p>

                    </div>

                </div>

            </div>

        </div>

    </section>

    <div class="modal fade" role="dialog" tabindex="-1" id="modal-1">

        <div class="modal-dialog" role="document">

            <div class="modal-content">

                <div class="modal-header">
```

```
                <h4 class="modal-title">American Sign Language - Alphabets</h4><button
type="button"

                    class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

            </div>

            <div class="modal-body"><img src="{{ url_for('static',
filename='img/ASL_Alphabets.png') }}" width="100%"></div>

            <div class="modal-footer"><button class="btn btn-secondary" type="button"

                    data-bs-dismiss="modal">Close</button></div>

        </div>

    </div>

</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></scri
pt>

</body>

</html>
```

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-34607-1660239739