V.S.B. ENGINEERING COLLEGE
(Affiliated by Anna University)
Karudayampalayam, Karur-639111

Real-Time Communication System Powered byAI for Specially Abled person

A Project report submitted in partial fulfilment of 7th semester in degree of

BACHELOR OF ENGINEERING
IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

TEAM ID: PNT2022TMID33352

TEAM LEADER : KAVIPRIYA G (922519104072)

TEAM MEMBER 1: MOHANAPRIYA C (922519104094)

TEAM MEMBER 2: MOULIKA P (922519104096)

TEAM MEMBER 3: KEERTHANA P T (922519104078)

CONTENTS

ABSTRACT:

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. New system that helps convert sign language to text and speech for easier communication with audience.

1.INTRODUCTION:

a. Project Overview:

Gesture is a non-verbal means of communication. It refers to expressing an idea using position, orientation or movement of a body part. Gesture recognition is the mathematical interpretation of orientation or motion of human body by a computational system. In this project, the words expressed by hand gestures by the speech and hearing impaired are converted into verbal means of communication. The translated output is displayed on a screen and "spoken" on a speaker.

Sign Language is the well-structured code, which uses hand gestures instead of sound to convey meaning, simultaneously combining hand shapes, orientations and movement of the hands. Communicative hand glove is an electronic device that can translate sign language into speech and text in order to make the communication possible between the deaf and/or mute with the general public. This technology has been used in a variety of application areas, which demands accurate interpretation of sign language. In this project, the words/letters conveyed by the disabled person are displayed on a screen and also spoken on a speaker.

b. Purpose:

The project aims to develop 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 using unconventional neural network.

An app is built which enables the deaf and dumb people conveytheir information using signs which is converted to human understandable language and output is given as speech.

2. LITERATURE SURVEY:

a. Existing problem:

Communications between deaf-mute and a normal person has alwaysbeen a challenging task. It is very difficult for mute people to conveytheir message to normal people. Since normal people are not trained on hand sign language. Only specially a bled people are taught sign language and the common person is unaware its working causing a communication gap. Under emergency situations, it is even more difficult for specially abled people to get help. Non-Emergency normal environments can also behard for them to navigate-needing special assistance.

b. References:

1. UFOs, K., EL Haloui, K., Dianati, M., Higgins, M., Elmirghani, J., Imran, M. A., & Tafazolli, R. (2021). Trends in Intelligent Communication Systems: Review of Standards, Major Research Projects, and Identification of Research Gaps. Journal of Sensor and Actuator Networks, 10(4), 60.ibm.com/blogs/internet-of-things/connected-trains-rail-travel/

2. Panda, G., Upadhyay, A. K., & Khandelwal, K. (2019). Artificial intelligence: A strategic disruption in public relations. Journal of Creative Communications, 14(3), 196-213.
3. Xu, G., Mu, Y., & Liu, J. (2017). Inclusion of artificial intelligence in communication networks and services. ITU J. ICT Discov. Speech, 1, 1-6.
4. Verma, P., Shimi S. L. and Priyadarshani, R., "Design of CommunicationInterpreter for Deaf and Dumb Person", Vol.4, no.1, 2013.

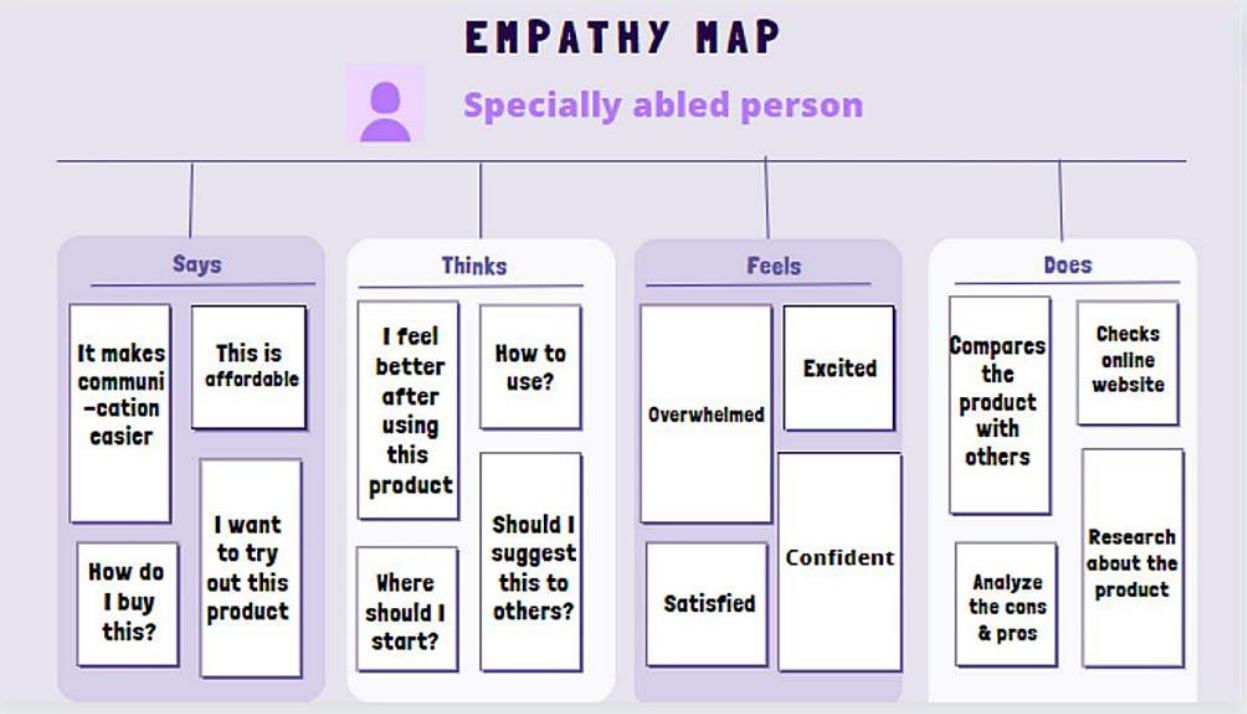
c. PROBLEM STATEMENT DEFINITION:

Only specially abled people are taught sign language and the common person is unaware its working causing a communication gap. Under emergency situations, it is even more difficult for specially abled people to get help. Non-Emergency normal environments can also behard for them to navigate needing special assistance.

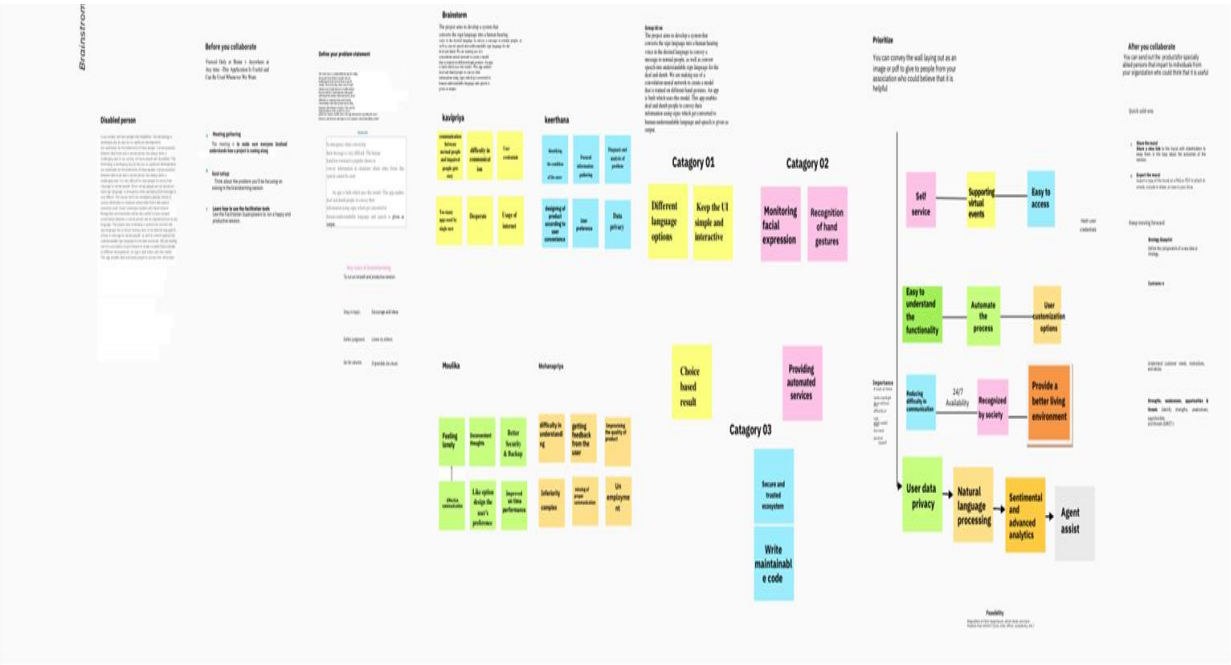
Communications between deaf-mute and a normal person has alwaysbeen a challenging task. It is very difficult for mute people to conveytheir message to normal people. Since normal people are not trained on hand sign language.

3. IDEATION &PROPOSED SOLUTION:

a. Empathy Map Canvas:



b. Ideation & Brainstorming:



c. 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.

d. Problem Solution fit:

4. REQUIREMENT ANALYSIS:

a. Functional requirement:

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Authentication	Confirmation through Facial acknowledgment Confirmation through secret kay verification convention
FR-4	External interfaces	Robots and different apparatuses give locally situated care also, other help, permitting individuals with handicaps to freely live
FR-5	Transaction processing	Many application can use to interpret the communication through signing like D talk in the framework
FR-6	Reporting	There is a developing indication that we want to accomplish more, to assist make the existences of individuals with handicaps more straightforward

B. Non-Functional requirements:

Non-functional Requirements:

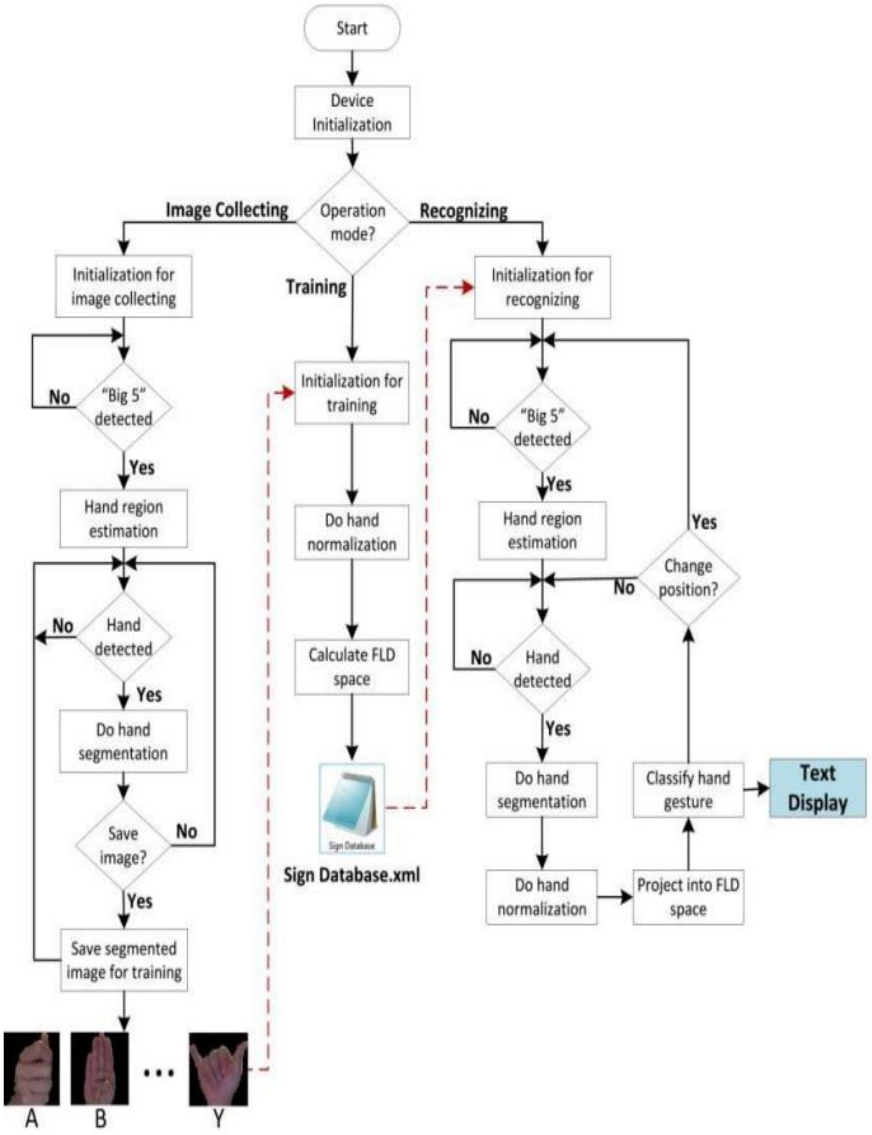
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The application and the product we are developing will be enabled with the facilities of voice engines which helps the user to analyse the surroundings and act accordingly. It also includes features such as speech to text and speech to signs and vice versa which allows deaf people to communicate with the outside world.
NFR-2	**Security**	This application provides a highly confidential platform when it comes to user security. It stores each and every detail of the users in a highly secured database which is impossible to access by the third parties.

5. PROJECT DESIGN:

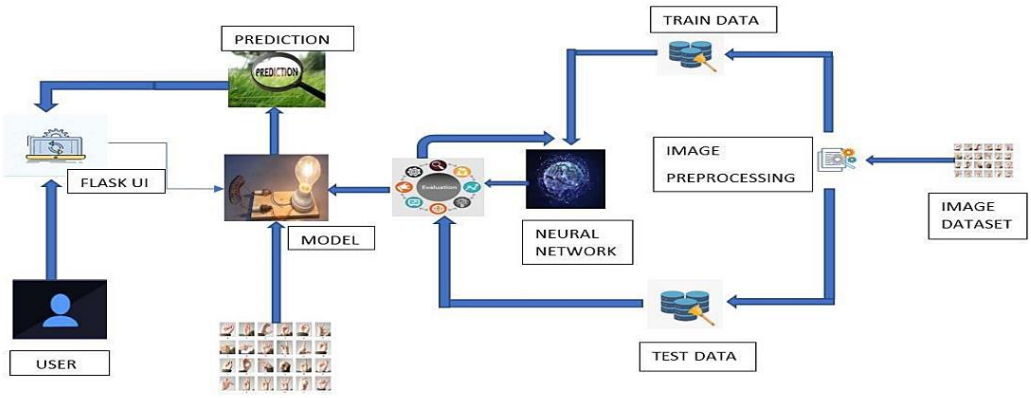
a. Data Flow Diagrams:

Dataflow Diagram:



b. Solution & Technical Architecture:

Technical Architecture:



c. User Stories:

User Stories:

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer *(Desktop user)*	Registration	USN-1	**Not Required**	I can access my account dashboard	High	Sprint-1
	Login	USN-2	**Not Required**		High	Sprint-1
	Dashboard	USN-3	**Not Required**			
Customer *(Desktop user)*	**Main page**	USN-4	As a User, **I can enter the web page once clicked, which provides be the Guidelines** to use the app	I can **enter the web page** once clicked	Medium	Sprint-1
Customer (Desktop user)	**Guidelines**	USN-5	As a User, **I can give a read through the guidelines to understand the functioning of the app.**	I can **give a read through the guidelines.**	Medium	Sprint-1
Customer *(Desktop user)*	**Convert Sign**	USN-6	As a User, **I can click the button** Convert sign , which directs me towards the Main screen	I can **click the button Convert sign and directed me to main screen.**	Medium	Sprint-2
Customer *(Desktop user)*	**Camera (Hand movement detection)**	USN-7	As a User, **I can show my hand sign towards the camera which converts them into text manner.**	**I can show my hand sign** towards the camera accurately.	High	Sprint-2
Customer *(Desktop user)*	**Voice mode**	USN-8	Once the text is obtained, As a User **I can click on the voice mode which provides the text in the form of speech.**	**I can click on the voice mode** which provides the text in the form of speech.	High	Sprint-2

Customer Care Executive	**Provide the necessary functionalities required to use the app.**		As an Executive, I can **provide the Specifications of Camera required, and other factors** that are required for smooth functioning of the app.	**I can provide the Specifications** of Camera required, and other factors	Low	Sprint-1
Customer Care Executive	**Check the performance of the app**		As an Executive, I can **check the usage and queries obtained from the end users**.	I can **check the usage and queries** obtained from the end users.	Medium	Sprint-1

6. PROJECT PLANNING& SCHEDULING:

a. Sprint Planning & Estimation:

Milestone Activity Plan

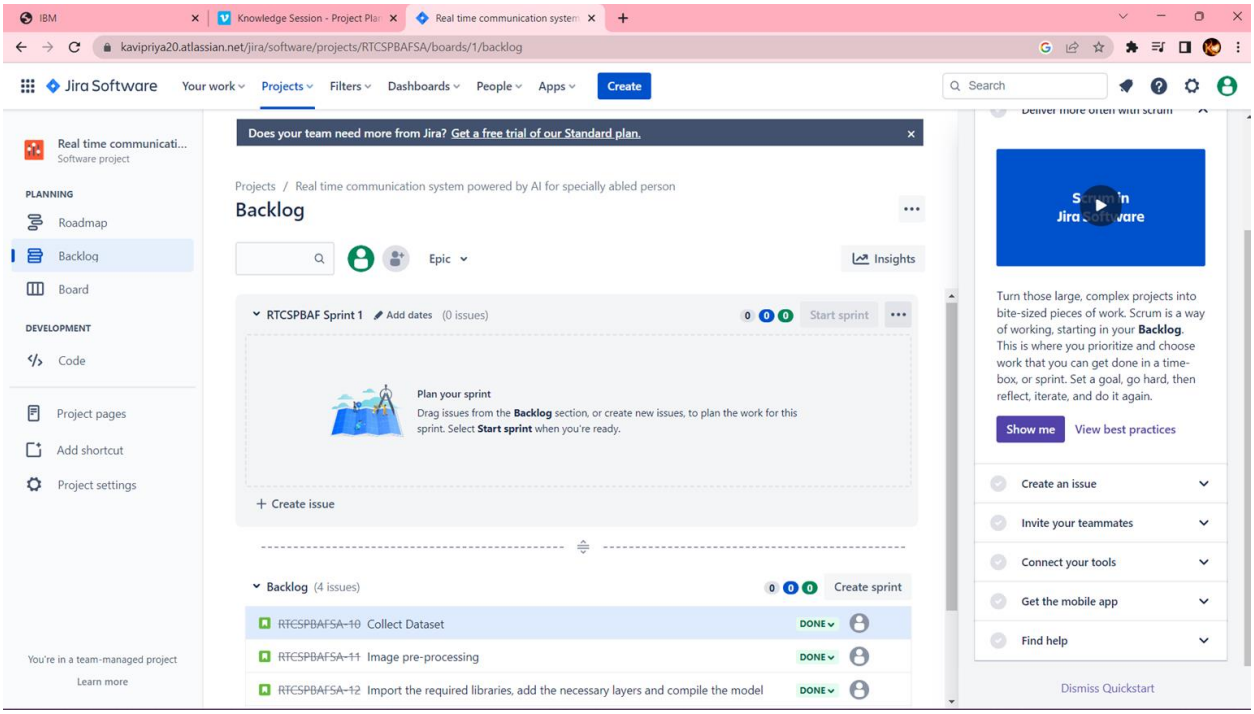
Milestone	Functional Requirement (Epic)	Milestone Story Number	Milestone Story / Task
Milestone 1	Data Collection	M1	We're collecting dataset for building our extend and making two organizers, one for preparing and another one for testing.
Milestone 2	Image Pre-processing	M2	Bringing in picture information generator libraries and applying picture information generator usefulness to prepare the test set.
Milestone 3	Building Model	M3	Bringing in the show building libraries, Initializing the show, Including Convolution layers, Including the Pooling layers, Including the Straighten layers, Including Thick layers, Compiling the demonstrate Fit and Spare the demonstrate.
Milestone 4	Testing Model	M4	Consequence the bundles to begin with. At that point we spare the demonstrate and stack the test picture, pre-process it and anticipate it.
Milestone 5	Application Layer	M5	Construct the flask application and the HTML pages.
Milestone 6	Train Conversation Engine	M6	Enrol for IBM Cloud and Train Picture Classification Demonstrate
Milestone 7	Final Result	M7	To guarantee all the exercises and coming about the ultimate yield.

b. Sprint Delivery Schedule:

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	Collect Dataset.	9	High	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-1		USN-2	Image pre-processing	8	Medium	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-2	Model Building	USN-3	Import the required libraries, add the necessary layers and compile the model	10	High	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-2		USN-4	Training the image classification model using CNN	7	Medium	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-3	Training and Testing	USN-5	Training the model and testing the model's performance	9	High	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-4	Implementation of the application	USN-6	Converting the input sign language images into English alphabets	8	Medium	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T

c. Report from JIRA:



7. CODING & SOLUTIONING (Explain the features added in the project along with code):

a. Model Building:

Model Building

Import The Required Model Building Libraries

In [ ]:
```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
#training datagen
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [ ]:
```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

In [ ]:
```
import tensorflow as tf
import os
```

Initialize The Model

In [ ]:
```
#create model
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping the dataset

In [ ]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

Model Building

Import The Required Model Building Libraries

In [ ]:
```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
#training datagen
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [ ]:
```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

In [ ]:
```
import tensorflow as tf
import os
```

Initialize The Model

In [ ]:
```
#create model
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping the dataset

In [ ]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

Applying ImageDataGenerator to training set

In [ ]:
```
x_train=train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

```
Found 15750 images belonging to 9 classes.
```

Applying ImageDataGenerator to test set

In [ ]:
```
x_test=test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

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

In [ ]:
```
a=len(x_train)
b=len(x_test)
```

Length of training set

In [ ]:
```
print(a)
```

```
79
```

Length of test set

In [ ]:
```
print(b)
```

```
12
```

Add Layers

In [ ]:
```
#create model
model=Sequential()
```

Import The Required Model Building Libraries

In [ ]:
```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
#training datagen
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [ ]:
```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

In [ ]:
```
import tensorflow as tf
import os
```

Initialize The Model

In [ ]:
```
#create model
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping the dataset

In [ ]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

Applying ImageDataGenerator to training set

In [ ]:
```
x_train=train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

```
Found 15750 images belonging to 9 classes.
```

Applying ImageDataGenerator to test set

In [ ]:
```
x_test=test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

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

In [ ]:
```
a=len(x_train)
b=len(x_test)
```

Length of training set

In [ ]:
```
print(a)
```

```
79
```

Length of test set

In [ ]:
```
print(b)
```

```
12
```

Add Layers

In [ ]:
```
#create model
model=Sequential()
```

Add The Convolution Layer

In [ ]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1),activation='relu'))
```

Model Building

Import The Required Model Building Libraries

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

In [2]:
```
#training datagen
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [3]:
```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

In [4]:
```
import tensorflow as tf
import os
```

Initialize The Model

In [5]:
```
#create model
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping the dataset

In [9]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

Applying ImageDataGenerator to training set

In [10]:
```
x_train=train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

```
Found 15750 images belonging to 9 classes.
```

Applying ImageDataGenerator to test set

In [11]:
```
x_test=test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=200,
                                        class_mode='categorical',color_mode="grayscale")
```

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

In [12]:
```
a=len(x_train)
b=len(x_test)
```

Length of training set

In [13]:
```
print(a)
```

```
79
```

Length of test set

In [14]:
```
print(b)
```

```
12
```

Add Layers

In [15]:
```
#create model
model=Sequential()
```

Add The Convolution Layer

In [16]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1),activation='relu'))
```

Add Pooling Layer

In [17]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

Import The Required Model Building Libraries

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

In [2]:
```
#training datagen
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [3]:
```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

In [4]:
```
import tensorflow as tf
import os
```

Initialize The Model

In [5]:
```
#create model
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping the dataset

In [9]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

```
Found 15750 images belonging to 9 classes.
```

Applying ImageDataGenerator to test set

In [11]:
```
x_test=test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=200,
                                        class_mode='categorical',color_mode="grayscale")
```

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

In [12]:
```
a=len(x_train)
b=len(x_test)
```

Length of training set

In [13]:
```
print(a)
```

```
79
```

Length of test set

In [14]:
```
print(b)
```

```
12
```

Add Layers

In [15]:
```
#create model
model=Sequential()
```

Add The Convolution Layer

In [16]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1),activation='relu'))
```

Add Pooling Layer

In [17]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

Add The Flatten Layer

In [18]:
```
model.add(Flatten())
```

Import The Required Model Building Libraries

In [ ]:
```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
#training datagen
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [ ]:
```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

In [ ]:
```
import tensorflow as tf
import os
```

Initialize The Model

In [ ]:
```
#create model
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping the dataset

In [ ]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

Applying ImageDataGenerator to training set

In [ ]:
```
x_train=train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

```
Found 15750 images belonging to 9 classes.
```

Applying ImageDataGenerator to test set

In [ ]:
```
x_test=test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

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

In [ ]:
```
a=len(x_train)
b=len(x_test)
```

Length of training set

In [ ]:
```
print(a)
```

```
79
```

Length of test set

In [ ]:
```
print(b)
```

```
12
```

Add Layers

In [ ]:
```
#create model
model=Sequential()
```

Add The Convolution Layer

In [ ]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1),activation='relu'))
```

Add Pooling Layer

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

Add The Flatten Layer

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

Adding The Dense Layers

In [ ]:
```
#1st hidden layer
model.add(Dense(units=512,activation='relu'))
#2nd hidden layer
model.add(Dense(units=261,activation='relu'))
```

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

Import The Required Model Building Libraries

In [ ]:
```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
#training datagen
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [ ]:
```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

In [ ]:
```
import tensorflow as tf
import os
```

Initialize The Model

In [ ]:
```
#create model
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping the dataset

In [ ]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

Applying ImageDataGenerator to training set

In [ ]:
```
x_train=train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

```
Found 15750 images belonging to 9 classes.
```

Applying ImageDataGenerator to test set

In [ ]:
```
x_test=test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

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

In [ ]:
```
a=len(x_train)
b=len(x_test)
```

Length of training set

In [ ]:
```
print(a)
```

```
79
```

Length of test set

In [ ]:
```
print(b)
```

```
12
```

Add Layers

In [ ]:
```
#create model
model=Sequential()
```

Add The Convolution Layer

In [ ]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1),activation='relu'))
```

Add Pooling Layer

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

Add The Flatten Layer

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

Adding The Dense Layers

In [ ]:
```
#1st hidden layer
model.add(Dense(units=512,activation='relu'))
#2nd hidden layer
model.add(Dense(units=261,activation='relu'))
```

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

Compile The Model

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

Import The Required Model Building Libraries

In [ ]:
```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
#training datagen
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [ ]:
```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

In [ ]:
```
import tensorflow as tf
import os
```

Initialize The Model

In [ ]:
```
#create model
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
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt #to view graph in colab itself
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping the dataset

In [ ]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

Applying ImageDataGenerator to training set

In [ ]:
```
x_train=train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

```
Found 15750 images belonging to 9 classes.
```

Applying ImageDataGenerator to test set

In [ ]:
```
x_test=test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=200,
                                          class_mode='categorical',color_mode="grayscale")
```

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

In [ ]:
```
a=len(x_train)
b=len(x_test)
```

Length of training set

In [ ]:
```
print(a)
```

```
79
```

Length of test set

In [ ]:
```
print(b)
```

```
12
```

Add Layers

In [ ]:
```
#create model
model=Sequential()
```

Add The Convolution Layer

In [ ]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1),activation='relu'))
```

Add Pooling Layer

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

Add The Flatten Layer

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

Adding The Dense Layers

In [ ]:
```
#1st hidden layer
model.add(Dense(units=512,activation='relu'))
#2nd hidden layer
model.add(Dense(units=261,activation='relu'))
```

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

Compile The Model

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

Fit The Mode

In [ ]:
```
model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versio
n. Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.
Epoch 1/10
79/79 [==============================] - 89s 1s/step - loss: 0.4318 - accuracy: 0.8572 - val_loss: 0.2326 - val_accuracy: 0.9471
Epoch 2/10
79/79 [==============================] - 87s 1s/step - loss: 0.0413 - accuracy: 0.9886 - val_loss: 0.1702 - val_accuracy: 0.9773
Epoch 3/10
79/79 [==============================] - 88s 1s/step - loss: 0.0253 - accuracy: 0.9933 - val_loss: 0.1599 - val_accuracy: 0.9764
Epoch 4/10
79/79 [==============================] - 86s 1s/step - loss: 0.0086 - accuracy: 0.9979 - val_loss: 0.1979 - val_accuracy: 0.9733
Epoch 5/10
79/79 [==============================] - 87s 1s/step - loss: 0.0097 - accuracy: 0.9975 - val_loss: 0.1815 - val_accuracy: 0.9782
Epoch 6/10
79/79 [==============================] - 86s 1s/step - loss: 0.0067 - accuracy: 0.9982 - val_loss: 0.2445 - val_accuracy: 0.9782
Epoch 7/10
79/79 [==============================] - 84s 1s/step - loss: 0.0045 - accuracy: 0.9988 - val_loss: 0.2291 - val_accuracy: 0.9782
Epoch 8/10
79/79 [==============================] - 84s 1s/step - loss: 0.0083 - accuracy: 0.9973 - val_loss: 0.1956 - val_accuracy: 0.9782
Epoch 9/10
79/79 [==============================] - 83s 1s/step - loss: 0.0029 - accuracy: 0.9995 - val_loss: 0.2011 - val_accuracy: 0.9773
Epoch 10/10
79/79 [==============================] - 84s 1s/step - loss: 0.0027 - accuracy: 0.9991 - val_loss: 0.2367 - val_accuracy: 0.9778
```

Out[ ]:

Save The Model

In [ ]:
```
model.save('aslpng2.h5')
```

8. TESTING:

a. Test Cases:

In [4]:
```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

In [5]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

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]:
```
import tensorflow as tf
import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
import IPython.display as display
from PIL import Image
import pathlib
```

Unzipping

In [6]:
```
!unzip '/content/drive/MyDrive/Colab Notebooks/conversation engine for deaf and dumb.zip'
```

b. User Acceptance Testing:

1.Purpose of Document:

The purpose of this document is to briefly explain the test coverage and open issues of the project at the time of the release to User Acceptance Testing (UAT).

2. 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	2	3	2	18
Duplicate	1	3	4	0	8
External	3	5	0	0	8
Fixed	12	2	5	22	41
Not Reproduced	0	1	0	0	1
Skipped	0	0	1	2	3
Won't Fix	0	4	1	1	7
Totals	27	17	14	27	86

Test Case Analysis:

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	49	0	0	49
Security	4	0	0	4
Outsource Shipping	4	0	0	4
Exception Reporting	11	0	0	11
Final Report Output	2	0	0	2
Version Control	1	0	0	1

9. RESULTS:

a. Performance Metrics:

PARAMETERS	FEATURES
Bandwidth	User can have access to this application even with low or reasonable internet bandwidth.
Offline	There is a mode called offline mode in this application. If the user enables the offline mode, he/she can able to access some specific features.
Emergency Ping	Emergency ping is a feature which enables the user to reach out the customer help line by just one ping. If the user is unable to type or use the application in an efficient manner can contact the application customer helpdesk by just clicking on the emergency ping button.

10.ADVANTAGES & DISADVANTAGES:

Advantages:

- Itis a cost-effective way of getting several people from different locations to attend meetings and conferences.
- It enables employees from across the world to communicate with each other 24×7 and share ideas or solve problems quickly.

Disadvantages:

- Also, accuracy depends upon distance between camera and object.
- It takes a lot of time to listen, speak, read, or write to someone.

11.CONCLUSION:

The proposed communication system between Deaf and Dumb peopleand ordinary people are aiming for it when bridging the communication gap between two societies. It provides complete two - sided communication in an efficient manner between the disabled andthe normal person.

For communication between deaf person and a second person, a mediator is required to translate sign language of deaf person. But a mediator is required to know the sign language used by deaf person. But this is not always possible since there are multiple sign languagesfor multiple languages.

So, to understand all sign languages, Hand gestures of deaf peoples bynormal peoples this system is proposed.

12.FUTURE SCOPE:

The speech-to-text and text-to-speech technologies helped those people who had difficulties in communicating or expressing theirfeelings to the normal people.

This reduces the communication gap between the normal people and the specially abled people.

Using image pre-processing and Artificial Intelligence it is easy to understand the context of objects and clearly explains it to the peoplewho use it for communication.

13. APPENDIX:

Source Code:

```
import cv2

video = cv2.VideoCapture(0)

while True:
    ret, frame = video.read()
    cv2.imshow("Frame", frame)
    k = cv2.waitKey(1)
    if k == ord('q'):
        break

video.release()
cv2.destroyAllWindows()
```

```python
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

class Video(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)
        self.roi_start = (50, 150)
        self.roi_end = (250, 350)
        self.model = load_model('asl_model.h5') # Execute Local Trained Model
        # self.model = load_model('IBM_Communication_Model.h5') # Execute IBM Trained Model
        self.index=['A','B','C','D','E','F','G','H','I']
        self.y = None
    def __del__(self):
        self.video.release()
    def get_frame(self):
        ret,frame = self.video.read()
        frame = cv2.resize(frame, (640, 480))
        copy = frame.copy()
        copy = copy[150:150+200,50:50+200]
        # Prediction Start
        cv2.imwrite('image.jpg',copy)
        copy_img = image.load_img('image.jpg', target_size=(64,64))
        x = image.img_to_array(copy_img)
        x = np.expand_dims(x, axis=0)
        pred = np.argmax(self.model.predict(x), axis=1)
        self.y = pred[0]
        cv2.putText(frame,'The Predicted Alphabet is: '+str(self.index[self.y]),(100,50),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),3)
        ret,jpg = cv2.imencode('.jpg', frame)
        return jpg.tobytes()
```

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial, Helvetica, sans-serif;}

/* Full-width input fields */
input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

/* Set a style for all buttons */
button {
  background-color: #273298;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
}

button:hover {
  opacity: 0.8;
}
```

```
/* Extra styles for the cancel button */
.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

/* Center the image and position the close button */
.imgcontainer {
```

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial, Helvetica, sans-serif;}

/* Full-width input fields */
input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

/* Set a style for all buttons */
button {
  background-color: #273298;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
}

button:hover {
  opacity: 0.8;
}
.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

/* Center the image and position the close button */
.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
  position: relative;
}

img.avatar {
  width: 40%;
  border-radius: 50%;
}

.container {
  padding: 16px;
}

span.psw {
  float: right;
  padding-top: 16px;
}

/* The Modal (background) */
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */
  left: 0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */
```

```
    overflow: auto; /* Enable scroll if needed */
    background-color: rgb(0,0,0); /* Fallback color */
    background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
    padding-top: 60px;
  }

  /* Modal Content/Box */
  .modal-content {
    background-color: #fefefe;
    margin: 5% auto 15% auto; /* 5% from the top, 15% from the bottom and centered */
    border: 1px solid #888;
    width: 80%; /* Could be more or less, depending on screen size */
  }

  /* The Close Button (x) */
  .close {
    position: absolute;
    right: 25px;
    top: 0;
    color: #000;
    font-size: 35px;
    font-weight: bold;
  }

  .close:hover,
  .close:focus {
    color: red;
    cursor: pointer;
  }

  /* Add Zoom Animation */
  .animate {
    -webkit-animation: animatezoom 0.6s;
    animation: animatezoom 0.6s
  }
```

```
@-webkit-keyframes animatezoom {
  from {-webkit-transform: scale(0)}
  to {-webkit-transform: scale(1)}
}

@keyframes animatezoom {
  from {transform: scale(0)}
  to {transform: scale(1)}
}
```

```
@-webkit-keyframes animatezoom {
  from {-webkit-transform: scale(0)}
  to {-webkit-transform: scale(1)}
}

@keyframes animatezoom {
  from {transform: scale(0)}
  to {transform: scale(1)}
}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
  span.psw {
     display: block;
     float: none;
  }
  .cancelbtn {
     width: 100%;
  }
}
</style>
</head>
<body>

<h2 style="text-align: center; color: rgba(1, 2, 2, 0.874);">REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED</h2>

<button onclick="document.getElementById('id01').style.display='block'">Login</button>

<div id="id01" class="modal">

  <form class="modal-content animate" action="/action_page.php" method="post">
    <div class="imgcontainer">
      <span onclick="document.getElementById('id01').style.display='none'" class="close" title="Close Modal">&times;</span>
      <img src="https://static.vecteezy.com/system/resources/thumbnails/007/407/996/small/user-icon-person-icon-client-symbol-login-head-sign-icon-design-vector.jpg">
    </div>
```

```
    <div class="container">
      <label for="uname"><b>Username</b></label>
      <input type="text" placeholder="Enter Username" name="uname" required>

      <label for="psw"><b>Password</b></label>
      <input type="password" placeholder="Enter Password" name="psw" required>

      <button type="submit">Login</button>
      <label>
        <input type="checkbox" checked="checked" name="remember"> Remember me
      </label>
    </div>

    <div class="container" style="background-color:#f1f1f1">
      <button type="button" onclick="document.getElementById('id01').style.display='none'" class="cancelbtn">Cancel</button>
      <span class="psw">Forgot <a href="#">password?</a></span>
    </div>
  </form>
</div>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
<div class="display-cover">
    <video autoplay></video>
    <canvas class="d-none"></canvas>

    <div class="video-options">
        <select name="" id="" class="custom-select">
```

```
            <option value="">Select camera</option>
        </select>
    </div>

    <img class="screenshot-image d-none" alt="">

    <div class="controls">
        <button class="btn btn-danger play" title="Play"><i data-feather="play-circle"></i></button>
        <button class="btn btn-info pause d-none" title="Pause"><i data-feather="pause"></i></button>
        <button class="btn btn-outline-success screenshot d-none" title="ScreenShot"><i data-feather="image"></i></button>
    </div>
</div>

<script src="https://unpkg.com/feather-icons"></script>
```

```
<style>
.screenshot-image {
    width: 150px;
    height: 90px;
    border-radius: 4px;
    border: 2px solid whitesmoke;
    box-shadow: 0 1px 2px 0 rgba(0, 0, 0, 0.1);
    position: absolute;
    bottom: 5px;
    left: 10px;
    background: white;
}

.display-cover {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 70%;
    margin: 5% auto;
    position: relative;
}

video {
    width: 100%;
    background: rgba(0, 0, 0, 0.2);
}

.video-options {
    position: absolute;
    left: 20px;
    top: 30px;
}

.controls {
    position: absolute;
    right: 20px;
    top: 20px;
```

```
        display: flex;
    }

    .controls > button {
        width: 45px;
        height: 45px;
        text-align: center;
        border-radius: 100%;
        margin: 0 6px;
        background: transparent;
    }

    .controls > button:hover svg {
        color: white !important;
    }

    @media (min-width: 300px) and (max-width: 400px) {
        .controls {
            flex-direction: column;
        }

        .controls button {
            margin: 5px 0 !important;
        }
    }

    .controls > button > svg {
        height: 20px;
        width: 18px;
        text-align: center;
        margin: 0 auto;
        padding: 0;
    }

    .controls button:nth-child(1) {
        border: 2px solid #1a12b3;
    }
```

```
// When the user clicks anywhere outside of the modal, close it
window.onclick = function(event) {
    if (event.target == modal) {
        modal.style.display = "none";
    }
}
feather.replace();

const controls = document.querySelector('.controls');
const cameraOptions = document.querySelector('.video-options>select');
const video = document.querySelector('video');
const canvas = document.querySelector('canvas');
const screenshotImage = document.querySelector('img');
const buttons = [...controls.querySelectorAll('button')];
let streamStarted = false;

const [play, pause, screenshot] = buttons;

const constraints = {
  video: {
    width: {
      min: 1280,
      ideal: 1920,
      max: 2560,
    },
    height: {
      min: 720,
      ideal: 1080,
      max: 1440
    },
  }
};
</script>
<script>
const getCameraSelection = async () => {
  const devices = await navigator.mediaDevices.enumerateDevices();
```

```
    const videoDevices = devices.filter(device => device.kind === 'videoinput');
    const options = videoDevices.map(videoDevice => {
      return `<option value="${videoDevice.deviceId}">${videoDevice.label}</option>`;
    });
    cameraOptions.innerHTML = options.join('');
  };

  </script>
  <script>

  play.onclick = () => {
    if (streamStarted) {
      video.play();
      play.classList.add('d-none');
      pause.classList.remove('d-none');
      return;
    }
    if ('mediaDevices' in navigator && navigator.mediaDevices.getUserMedia) {
      const updatedConstraints = {
        ...constraints,
        deviceId: {
          exact: cameraOptions.value
        }
      };
      startStream(updatedConstraints);
    }
  };

  const startStream = async (constraints) => {
    const stream = await navigator.mediaDevices.getUserMedia(constraints);
    handleStream(stream);
  };

  const handleStream = (stream) => {
    video.srcObject = stream;
    play.classList.add('d-none');
    pause.classList.remove('d-none');
```

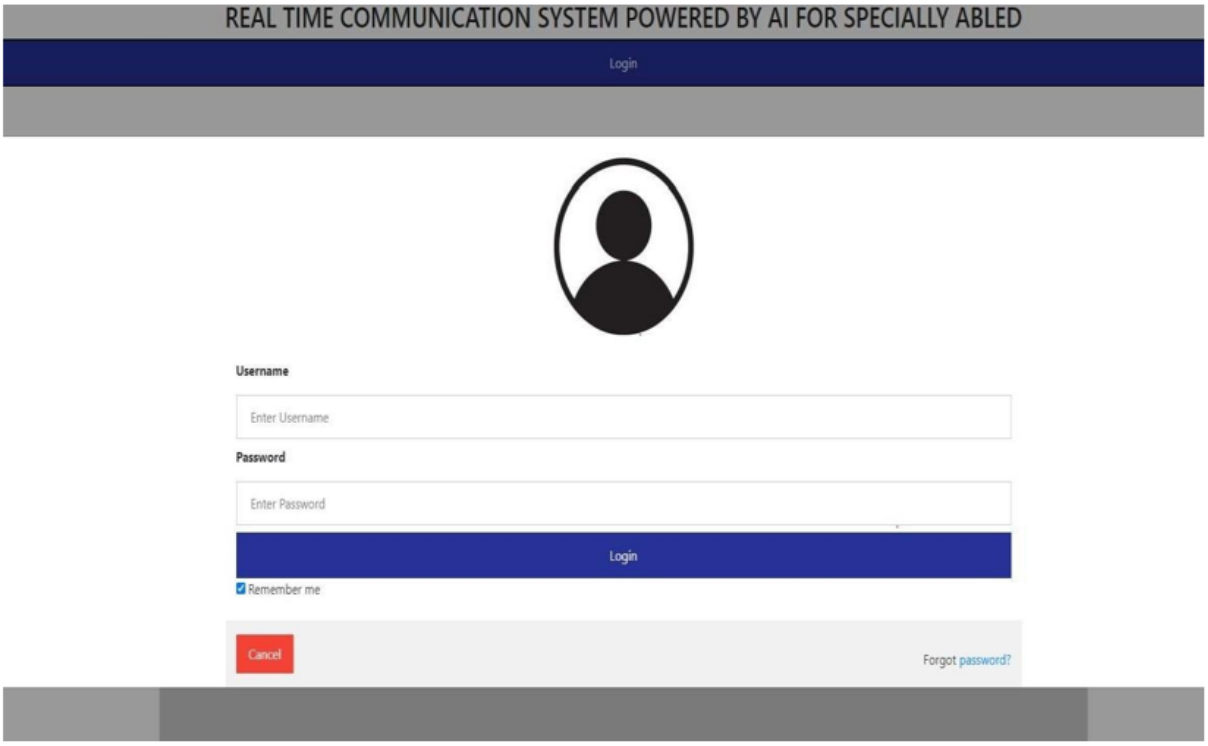
```
    screenshot.classList.remove('d-none');
    streamStarted = true;
  };

  getCameraSelection();
  ...
  cameraOptions.onchange = () => {
    const updatedConstraints = {
      ...constraints,
      deviceId: {
        exact: cameraOptions.value
      }
    };
    startStream(updatedConstraints);
  };

  const pauseStream = () => {
    video.pause();
    play.classList.remove('d-none');
    pause.classList.add('d-none');
  };

  const doScreenshot = () => {
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    canvas.getContext('2d').drawImage(video, 0, 0);
    screenshotImage.src = canvas.toDataURL('image/webp');
    screenshotImage.classList.remove('d-none');
  };

  pause.onclick = pauseStream;
  screenshot.onclick = doScreenshot;
  </script>

  </body>
  </html>
```

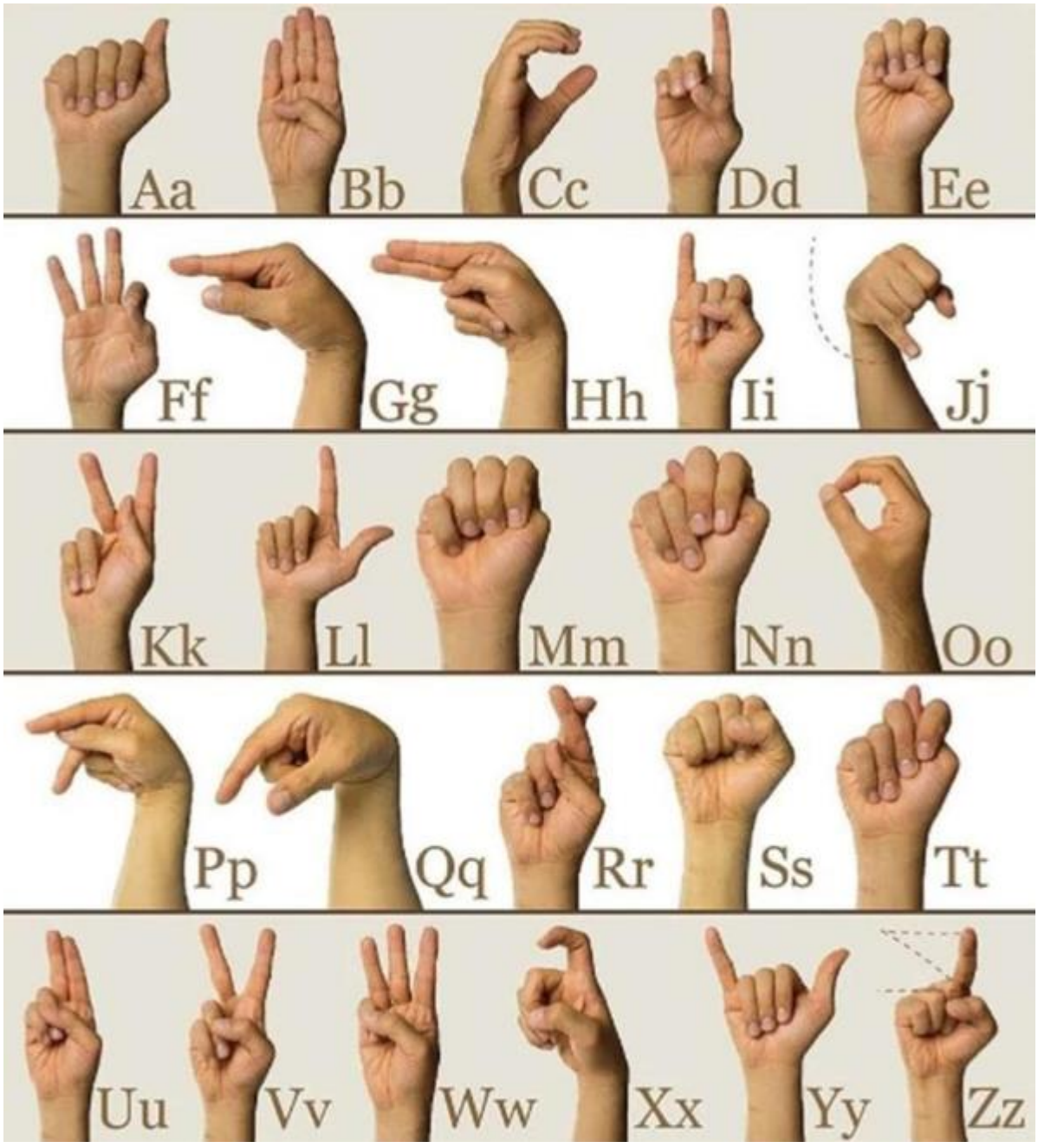
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
    <title>SmartBridge_WebApp_VideoTemplate</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="assets/css/Banner-Heading-Image.css">
    <link rel="stylesheet" href="assets/css/Navbar-Centered-Brand.css">
    <link rel="stylesheet" href="assets/css/styles.css">
</head>

<body style="background: rgb(39,43,48);">
    <nav class="navbar navbar-light navbar-expand-md py-3" style="background: #212529;">
        <div class="container">
            <div></div><a class="navbar-brand d-flex align-items-center" href="#"><span
                    class="bs-icon-sm bs-icon-rounded bs-icon-primary d-flex justify-content-center align-items-center me-2 bs-icon"><i
                        class="fas fa-flask"></i></span><span style="color: rgb(255,255,255);">Real-Time Communication
                    System Powered By AI&amp;nbsp;For Specially Abled</span></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: 640px;height: 480px;margin: 10px;min-height: 480px;min-width: 640px;border-radius: 10px;border: 4px dashed rgb(255,255,255) ;">
                <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>
        <div class="d-flex flex-column justify-content-center align-items-center" style="margin-bottom: 10px;"><button
                class="btn btn-info" type="button" data-bs-target="#modal-1" data-bs-toggle="modal">Quick Reference
                -<strong> ASL Alphabets</strong></button></div>
    </section>
    <section>
```

```
        <div class="container">
            <div class="accordion text-white" role="tablist" id="accordion-1">
                <div class="accordion-item" style="background: rgb(33,37,41);">
                    <h2 class="accordion-header" role="tab"><button class="accordion-button" data-bs-toggle="collapse"
                            data-bs-target="#accordion-1 .item-1" aria-expanded="true"
                            aria-controls="accordion-1 .item-1"
                            style="background: rgb(39,43,48);color: rgb(255,255,255);">About The Project</button></h2>
                    <div class="accordion-collapse collapse show item-1" role="tabpanel" data-bs-parent="#accordion-1">
                        <div class="accordion-body">
                            <p class="mb-0">Artificial Intelligence has made it possible to handle our daily activities
                                in new and simpler ways. With the ability to automate tasks that normally require human
                                intelligence, such as speech and voice recognition, visual perception, predictive text
                                functionality, decision-making, and a variety of other tasks, AI can assist people with
                                disabilities by significantly improving their ability to get around and participate in
                                daily activities.<br><br>Currently, Sign Recognition is available <strong>only for
                                    alphabets A-I</strong> and not for J-Z, since J-Z alphabets also require Gesture
                                Recognition for them to be able to be predicted correctly to a certain degree of
                                accuracy.</p>
                        </div>
                    </div>
                </div>
                <div class="accordion-item" style="background: rgb(33,37,41);">
                    <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="background: rgb(39,43,48);color: rgb(231,241,255);">Developed By</button></h2>
                    <div class="accordion-collapse collapse item-2" role="tabpanel" data-bs-parent="#accordion-1">
                        <div class="accordion-body">
                            <p class="mb-0">Students at VIT-Bhopal University during SmartBridge AI Externship
                                Program.<br><br>1. <strong>Nirlov Deb</strong> 19BCG10067<br>2.
                                <strong>Kushagra</strong> 19BCG10025<br>3. <strong>Kartik Dhasmana</strong> 19BCG10002
                            </p>
                        </div>
                    </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"></script>
</body>

</html>
```

```css
.bs-icon {
  --bs-icon-size: .75rem;
  display: flex;
  flex-shrink: 0;
  justify-content: center;
  align-items: center;
  font-size: var(--bs-icon-size);
  width: calc(var(--bs-icon-size) * 2);
  height: calc(var(--bs-icon-size) * 2);
  color: var(--bs-primary);
}

.bs-icon-xs {
  --bs-icon-size: 1rem;
  width: calc(var(--bs-icon-size) * 1.5);
  height: calc(var(--bs-icon-size) * 1.5);
}

.bs-icon-sm {
  --bs-icon-size: 1rem;
}

.bs-icon-md {
  --bs-icon-size: 1.5rem;
}

.bs-icon-lg {
  --bs-icon-size: 2rem;
}

.bs-icon-xl {
  --bs-icon-size: 2.5rem;
}

.bs-icon.bs-icon-primary {
  color: var(--bs-white);
  background: var(--bs-primary);
```

```
}

.bs-icon.bs-icon-primary-light {
  color: var(--bs-primary);
  background: rgba(var(--bs-primary-rgb), .2);
}

.bs-icon.bs-icon-semi-white {
  color: var(--bs-primary);
  background: rgba(255, 255, 255, .5);
}

.bs-icon.bs-icon-rounded {
  border-radius: .5rem;
}

.bs-icon.bs-icon-circle {
  border-radius: 50%;
}
```




Aa
Bb
Cc
Dd
Ee
Ff
Gg
Hh
Ii
Jj
Kk
Ll
Mm
Nn
Oo
Pp
Qq
Rr
Ss
Tt
Uu
Vv
Ww
Xx
Yy
Zz

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-38423-1660380326