REAL -TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYNMENT AND ENTERPRENEURSHIP

A PROJECT REPORT

TEAM ID – PNT2022TMID27680

PAUL TAMIL SELVAN J	**-**	**311419104063**
MOHAN RAJ S	**-**	**311419104058**
RUPESH DP	**-**	**311419104501**
MADHAN P	**-**	**311419104051**

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

MEENAKSHI COLLEGE OF ENGINEERING
CHENNAI – 600078

1. INTRODUCTION

Overview

People get to know one another by sharing their ideas, thoughts, and experiences with those around them. There are numerous ways to accomplish this, the best of which is the gift of "Speech." Everyone can very convincingly transfer their thoughts and understand each other through speech. It will be unjust if we overlookthose who are denied this priceless gift: the deaf and dumb. In such cases, the human hand has remained the preferred method of communication.

Purpose

The project's purpose is to create a system that translates sign language into a human-understandable language so that ordinary people may understand it.

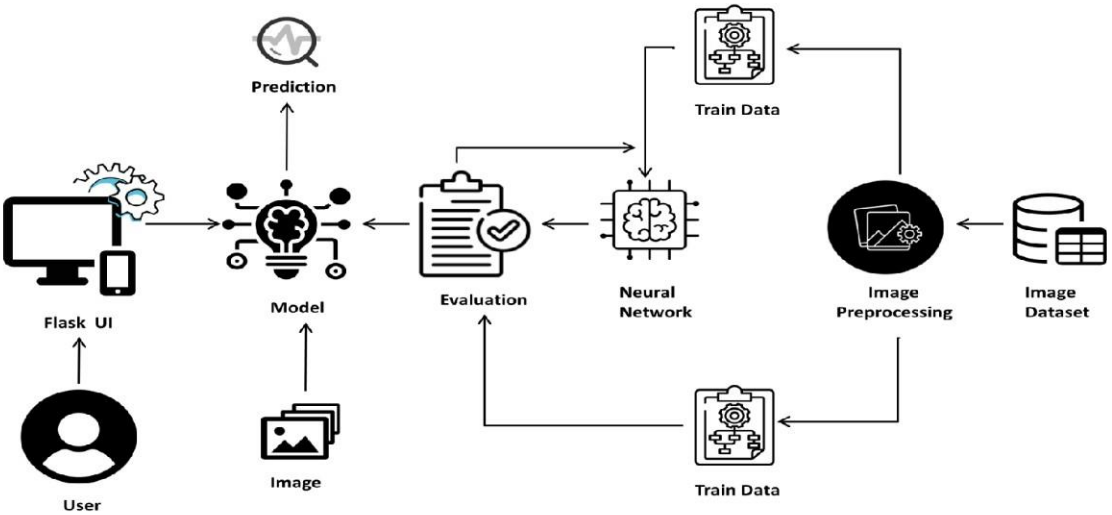
2. LITERATURE SURVEY

S.NO	TITLE	AUTHOR	YEAR& PUBLICATIONS	REMARKS
1.	Sign language to speech conversion	P. Vijayalakshmi and M.Aarthi	2016 International Conference on Recent Trends in Information Technology (ICRTIT)	They have designed a sensor-based gesture recognition module that recognizes English alphabets.
2.	Conversion of Sign Language into Text	Mahesh Kumar N B	International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 9 (2018)	In this model, the Linear Discriminant Analysis (LDA) algorithm was used for gesture recognition and recognized gestureis converted into text and voice format.
3.	Real time conversion of signlanguage to speech and prediction of gestures using Artificial Neural Network	Mahesh Kumar N BAbey Abraham, VRohini	Procedia Computer Science, Volume 143, 2018	The proposed device makes use of an Arduino Uno board, a few flex sensors and an Android application to enable effective communication amongst the users.

4.	Automated Speech to Sign language Conversion usingGoogle API and NLP	Bharti, Ritika and Yadav, Sarthak and Gupta, Sourav and B, Rajitha	Proceedings of the International Conference on Advances in Electronics, Electrical& Computational Intelligence (ICAEEC)2019	The proposed system first recognizes the speech, the second converts it to text, third matches tokenized text with the visual sign word library (videos of sign language), fourth concatenates all the matched videos according to the text recognized and finally display the merged video to the deaf/dumb person.
5.	Virtual assistant using python	Damarla.k	2021	To help physically impaired people we came up with an idea of erecting a voice adjunct with continues commerce point.

3. THEORITICAL ANALYSIS

Block diagram



Hardware / Software designing

Hardware Requirements:

Operating System	Windows, Mac, Linux
CPU (for training)	Multi Core Processors (i3 or above/equivalent)
GPU (for training)	NVIDIA AI Capable / Google's TPU
WebCam	Integrated or External with FullHD Support

Software Requirements:

Python	v3.9.0 or Above
Python Packages	flask, tensorflow, opencv-python, keras, numpy, pandas, virtualenv, pillow
Web Browser	Mozilla Firefox, Google Chrome or any modern web browser
IBM Cloud (for training)	Watson Studio - Model Training & Deployment as Machine Learning Instance

4. EXPERIMENTAL INVESTIGATIONS

Training and Testing using Dataset Provided

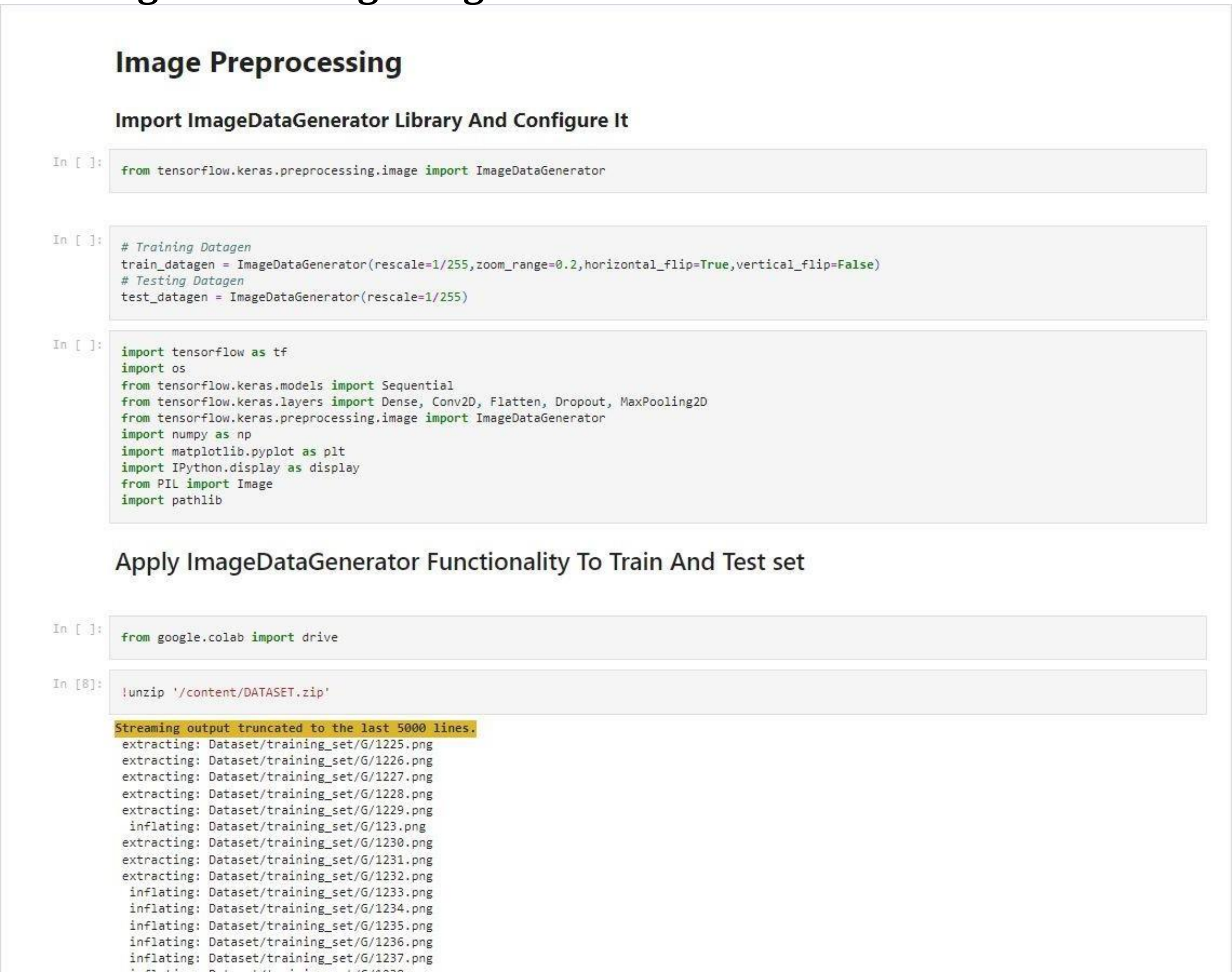


Image Preprocessing

Import ImageDataGenerator Library And Configure It

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

```
In [ ]:
# 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)
```

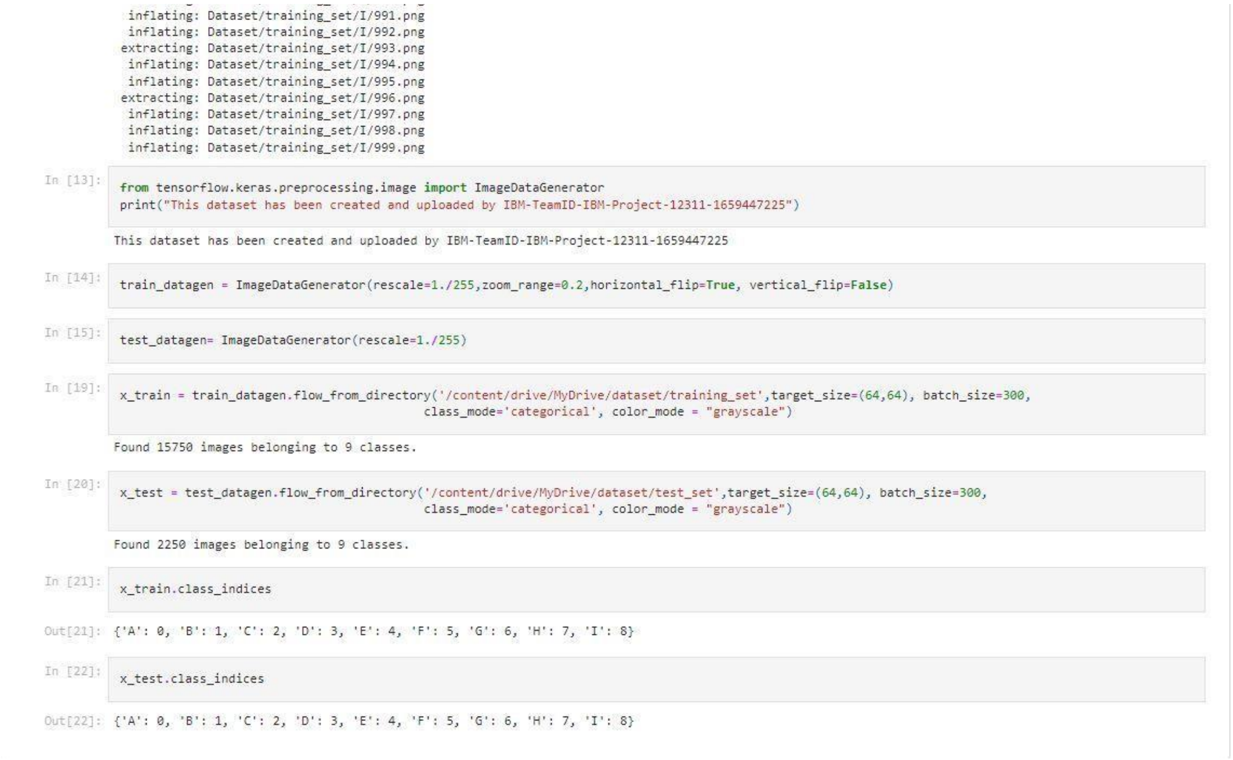
```
In [ ]:
import tensorflow as tf
import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
import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

Apply ImageDataGenerator Functionality To Train And Test set

```
In [ ]:
from google.colab import drive
```

```
In [8]:
!unzip '/content/DATASET.zip'
```

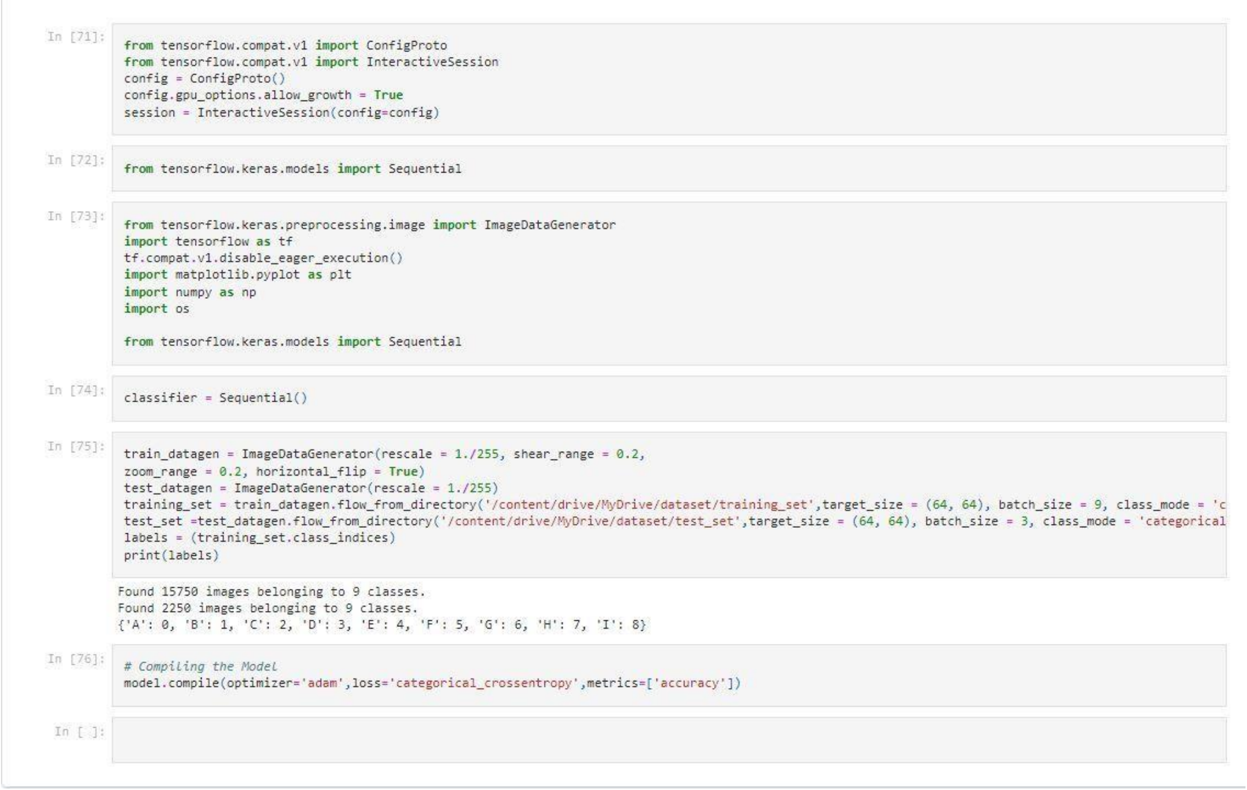
```
Streaming output truncated to the last 5000 lines.
 extracting: Dataset/training_set/G/1225.png
 extracting: Dataset/training_set/G/1226.png
 extracting: Dataset/training_set/G/1227.png
 extracting: Dataset/training_set/G/1228.png
 extracting: Dataset/training_set/G/1229.png
  inflating: Dataset/training_set/G/123.png
 extracting: Dataset/training_set/G/1230.png
 extracting: Dataset/training_set/G/1231.png
 extracting: Dataset/training_set/G/1232.png
  inflating: Dataset/training_set/G/1233.png
  inflating: Dataset/training_set/G/1234.png
  inflating: Dataset/training_set/G/1235.png
  inflating: Dataset/training_set/G/1236.png
  inflating: Dataset/training_set/G/1237.png
```



```
  inflating: Dataset/training_set/I/991.png
  inflating: Dataset/training_set/I/992.png
 extracting: Dataset/training_set/I/993.png
  inflating: Dataset/training_set/I/994.png
  inflating: Dataset/training_set/I/995.png
 extracting: Dataset/training_set/I/996.png
  inflating: Dataset/training_set/I/997.png
  inflating: Dataset/training_set/I/998.png
  inflating: Dataset/training_set/I/999.png
```

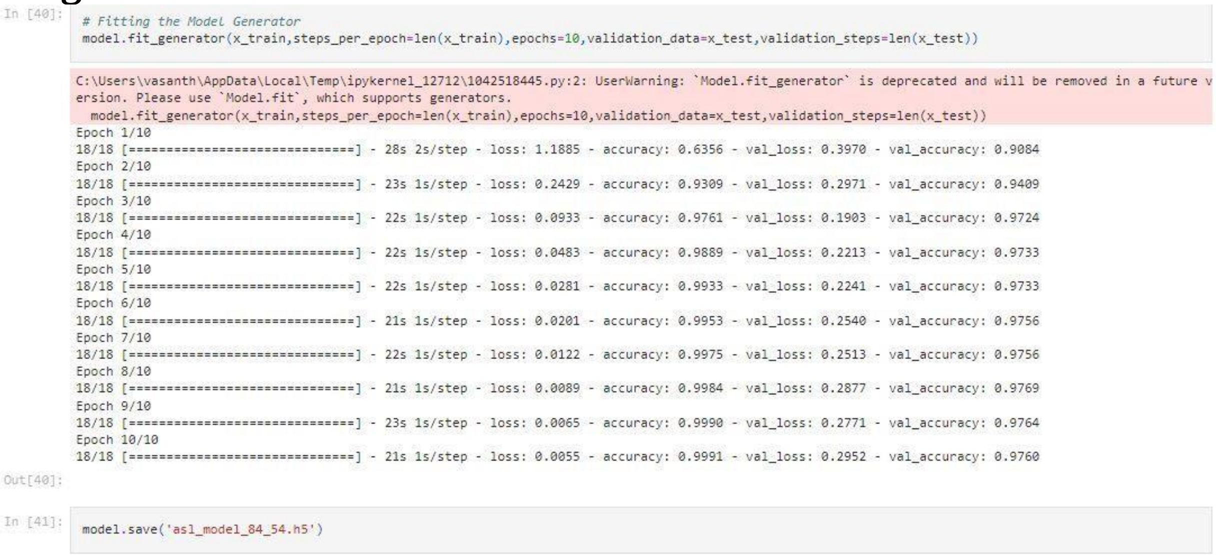
```
In [13]:  from tensorflow.keras.preprocessing.image import ImageDataGenerator
          print("This dataset has been created and uploaded by IBM-TeamID-IBM-Project-12311-1659447225")

         This dataset has been created and uploaded by IBM-TeamID-IBM-Project-12311-1659447225

In [14]:  train_datagen = ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True, vertical_flip=False)

In [15]:  test_datagen= ImageDataGenerator(rescale=1./255)

In [19]:  x_train = train_datagen.flow_from_directory('/content/drive/MyDrive/dataset/training_set',target_size=(64,64), batch_size=300,
                                                 class_mode='categorical', color_mode = "grayscale")

         Found 15750 images belonging to 9 classes.

In [20]:  x_test = test_datagen.flow_from_directory('/content/drive/MyDrive/dataset/test_set',target_size=(64,64), batch_size=300,
                                                 class_mode='categorical', color_mode = "grayscale")

         Found 2250 images belonging to 9 classes.

In [21]:  x_train.class_indices

Out[21]: {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}

In [22]:  x_test.class_indices

Out[22]: {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation



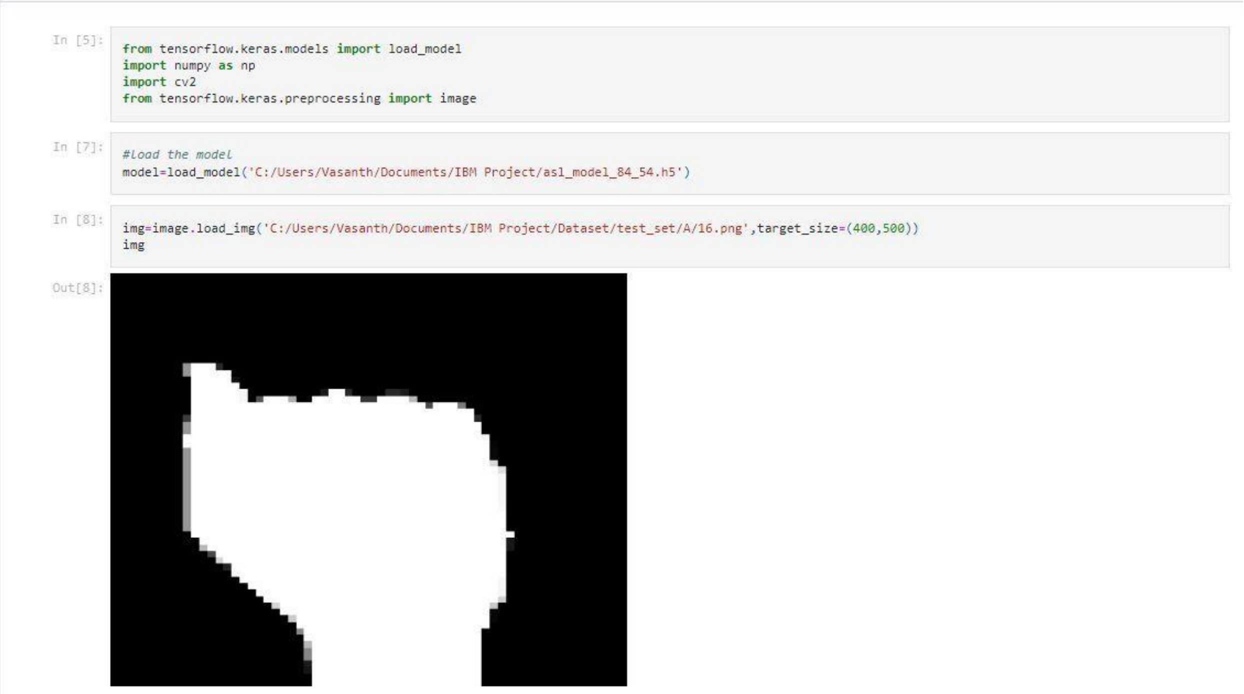
```
In [71]:  from tensorflow.compat.v1 import ConfigProto
          from tensorflow.compat.v1 import InteractiveSession
          config = ConfigProto()
          config.gpu_options.allow_growth = True
          session = InteractiveSession(config=config)

In [72]:  from tensorflow.keras.models import Sequential

In [73]:  from tensorflow.keras.preprocessing.image import ImageDataGenerator
          import tensorflow as tf
          tf.compat.v1.disable_eager_execution()
          import matplotlib.pyplot as plt
          import numpy as np
          import os

          from tensorflow.keras.models import Sequential

In [74]:  classifier = Sequential()

In [75]:  train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2,
          zoom_range = 0.2, horizontal_flip = True)
          test_datagen = ImageDataGenerator(rescale = 1./255)
          training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/dataset/training_set',target_size = (64, 64), batch_size = 9, class_mode = 'c
          test_set =test_datagen.flow_from_directory('/content/drive/MyDrive/dataset/test_set',target_size = (64, 64), batch_size = 3, class_mode = 'categorical
          labels = (training_set.class_indices)
          print(labels)

         Found 15750 images belonging to 9 classes.
         Found 2250 images belonging to 9 classes.
         {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}

In [76]:  # Compiling the Model
          model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

In [ ]:
```

Saving the Model



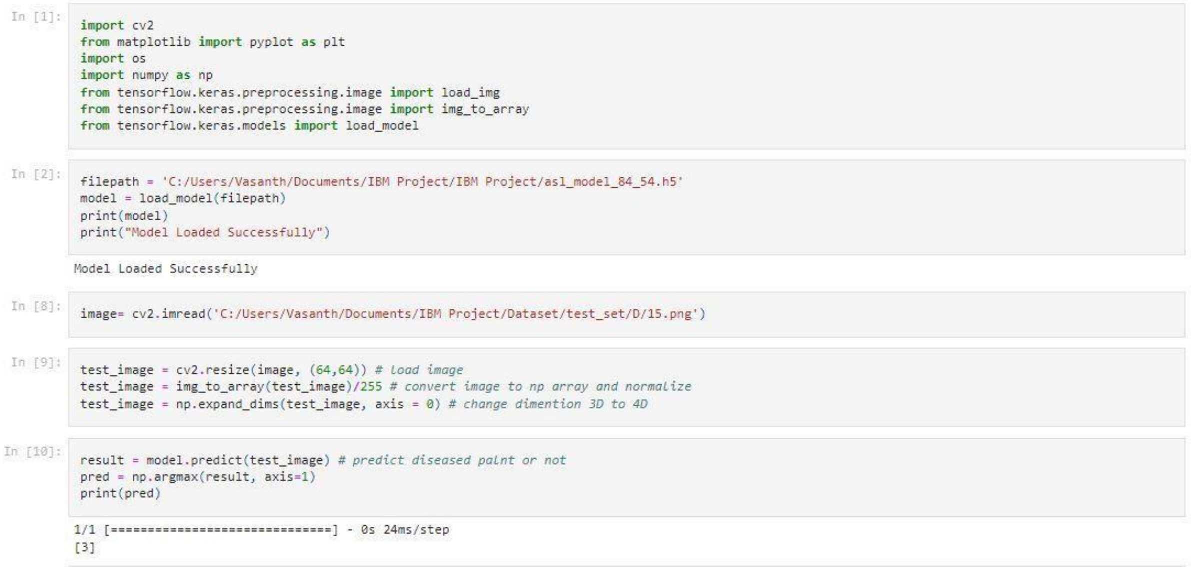
```
In [40]:  # Fitting the Model Generator
          model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))

          C:\Users\vasanth\AppData\Local\Temp\ipykernel_12712\1042518445.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future v
          ersion. Please use `Model.fit`, which supports generators.
            model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
          Epoch 1/10
          18/18 [==============================] - 28s 2s/step - loss: 1.1885 - accuracy: 0.6356 - val_loss: 0.3970 - val_accuracy: 0.9084
          Epoch 2/10
          18/18 [==============================] - 23s 1s/step - loss: 0.2429 - accuracy: 0.9309 - val_loss: 0.2971 - val_accuracy: 0.9409
          Epoch 3/10
          18/18 [==============================] - 22s 1s/step - loss: 0.0933 - accuracy: 0.9761 - val_loss: 0.1903 - val_accuracy: 0.9724
          Epoch 4/10
          18/18 [==============================] - 22s 1s/step - loss: 0.0483 - accuracy: 0.9889 - val_loss: 0.2213 - val_accuracy: 0.9733
          Epoch 5/10
          18/18 [==============================] - 22s 1s/step - loss: 0.0281 - accuracy: 0.9933 - val_loss: 0.2241 - val_accuracy: 0.9733
          Epoch 6/10
          18/18 [==============================] - 21s 1s/step - loss: 0.0201 - accuracy: 0.9953 - val_loss: 0.2540 - val_accuracy: 0.9756
          Epoch 7/10
          18/18 [==============================] - 22s 1s/step - loss: 0.0122 - accuracy: 0.9975 - val_loss: 0.2513 - val_accuracy: 0.9756
          Epoch 8/10
          18/18 [==============================] - 21s 1s/step - loss: 0.0089 - accuracy: 0.9984 - val_loss: 0.2877 - val_accuracy: 0.9769
          Epoch 9/10
          18/18 [==============================] - 23s 1s/step - loss: 0.0065 - accuracy: 0.9990 - val_loss: 0.2771 - val_accuracy: 0.9764
          Epoch 10/10
          18/18 [==============================] - 21s 1s/step - loss: 0.0055 - accuracy: 0.9991 - val_loss: 0.2952 - val_accuracy: 0.9760
Out[40]:

In [41]:  model.save('asl_model_84_54.h5')
```

Testing the Model



```
In [5]:  from tensorflow.keras.models import load_model
         import numpy as np
         import cv2
         from tensorflow.keras.preprocessing import image

In [7]:  #Load the model
         model=load_model('C:/Users/Vasanth/Documents/IBM Project/asl_model_84_54.h5')

In [8]:  img=image.load_img('C:/Users/Vasanth/Documents/IBM Project/Dataset/test_set/A/16.png',target_size=(400,500))
         img

Out[8]:
```

Predicting the model



```
In [1]:
import cv2
from matplotlib import pyplot as plt
import os
import numpy as np
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
```

```
In [2]:
filepath = 'C:/Users/Vasanth/Documents/IBM Project/IBM Project/asl_model_84_54.h5'
model = load_model(filepath)
print(model)
print("Model Loaded Successfully")
```

```
Model Loaded Successfully
```

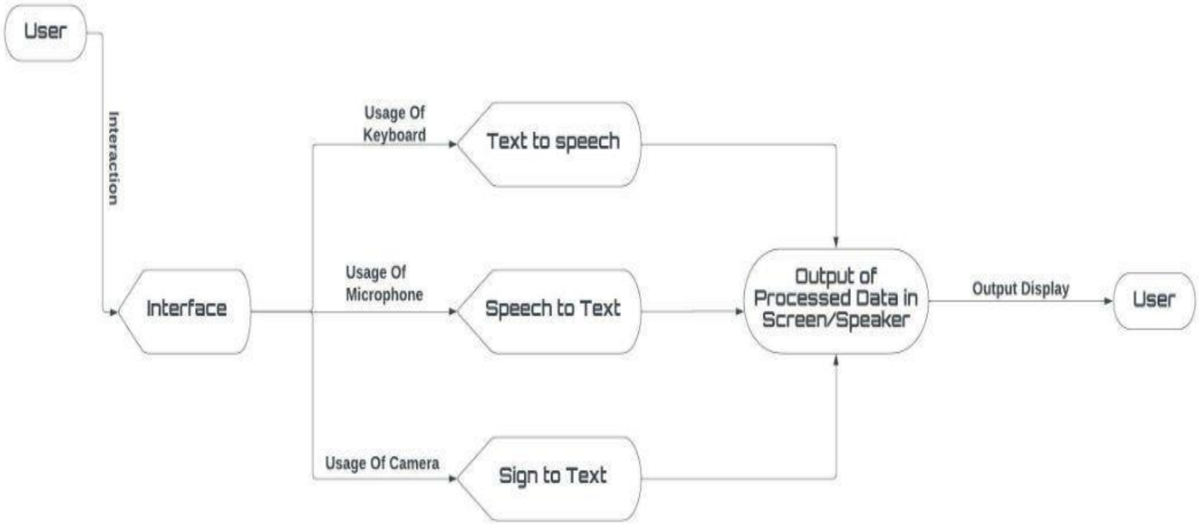
```
In [8]:
image= cv2.imread('C:/Users/Vasanth/Documents/IBM Project/Dataset/test_set/D/15.png')
```

```
In [9]:
test_image = cv2.resize(image, (64,64)) # Load image
test_image = img_to_array(test_image)/255 # convert image to np array and normalize
test_image = np.expand_dims(test_image, axis = 0) # change dimention 3D to 4D
```

```
In [10]:
result = model.predict(test_image) # predict diseased palnt or not
pred = np.argmax(result, axis=1)
print(pred)
```

```
1/1 [==============================] - 0s 24ms/step
[3]
```

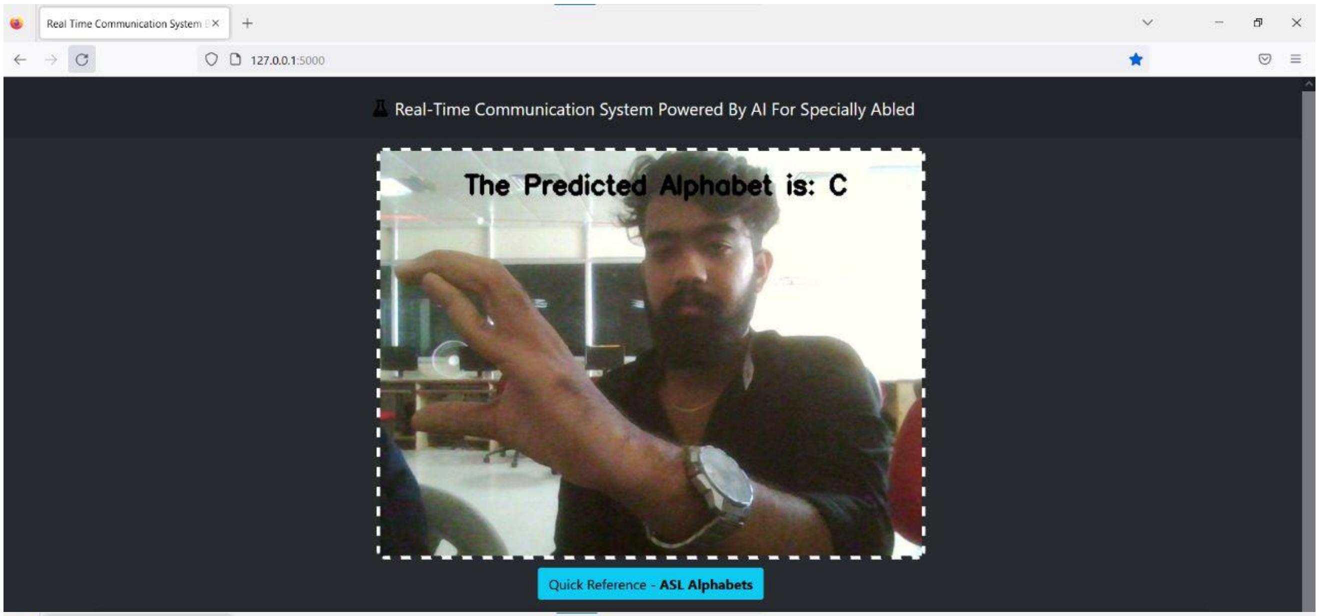
5. FLOWCHART



6. RESULT

The proposed procedure was implemented and tested with set of images. The set of 15750 images of Alphabets from “A” to “I” are used for training database and a set of 2250 images of Alphabets from “A” to “I” are used for testing database.Once the gesture is recognise the equivalent Alphabet is shown on the screen.

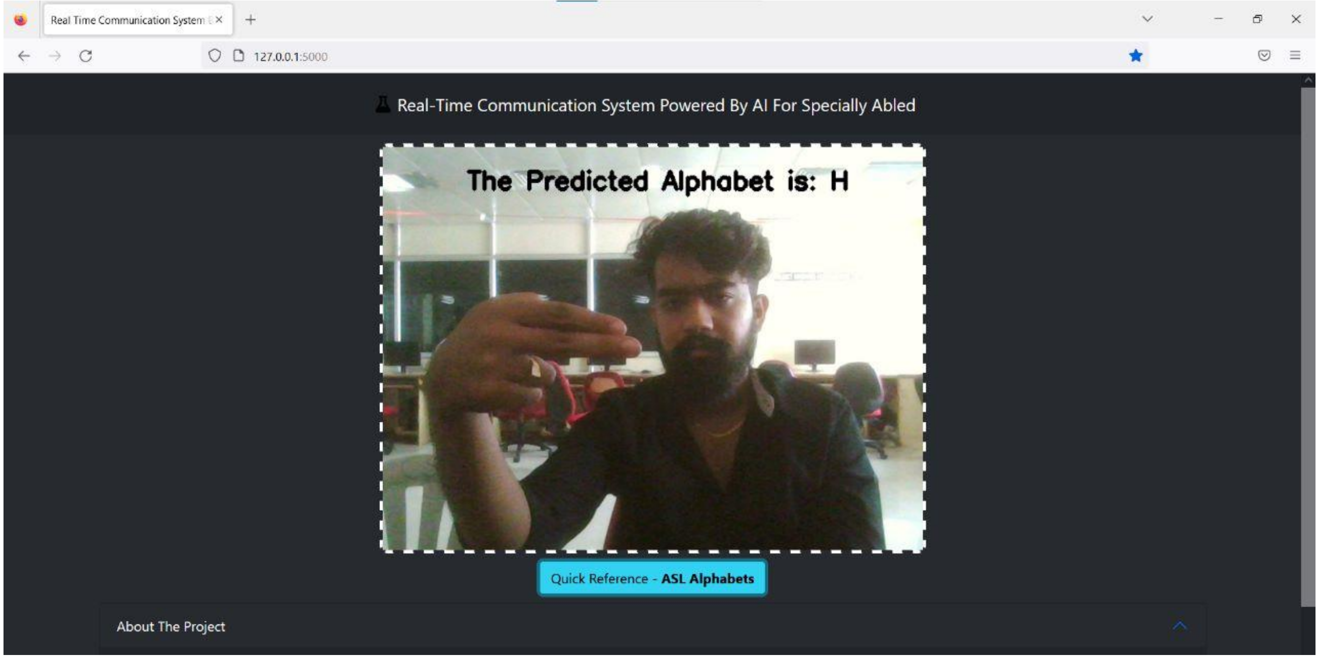
Some sample images of the output are provided below:


Real Time Communication System
127.0.0.1:5000

Real-Time Communication System Powered By AI For Specially Abled

The Predicted Alphabet is: C

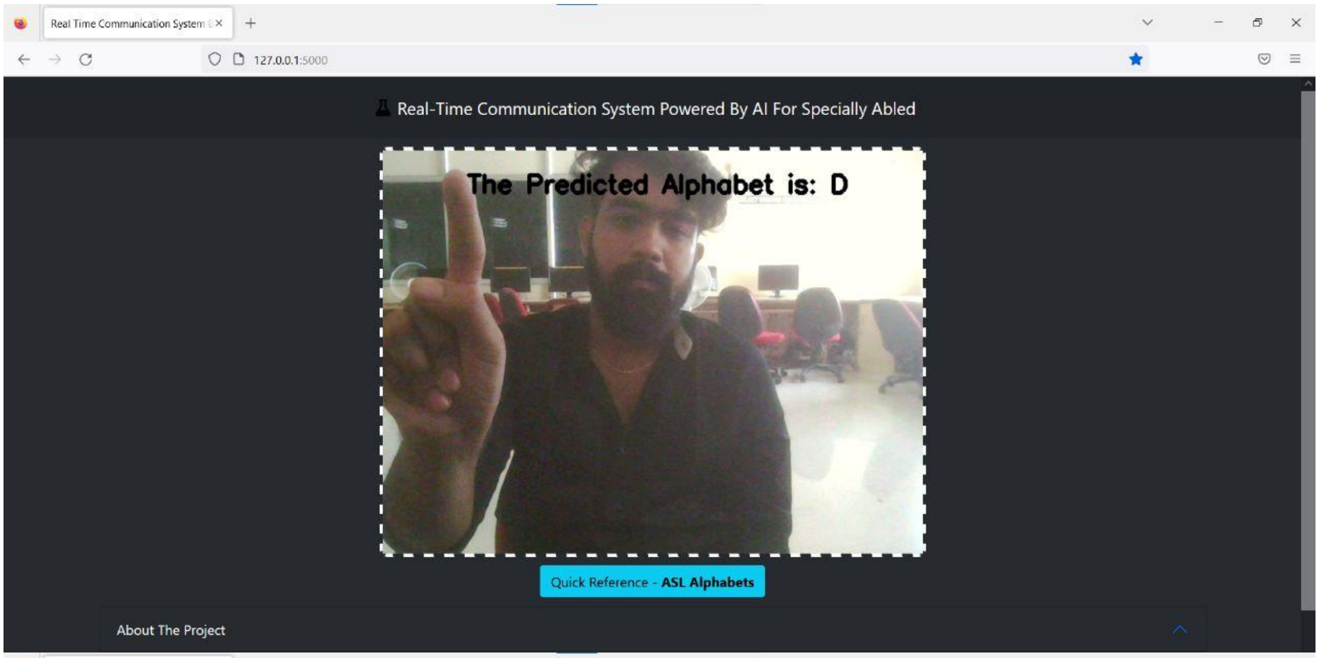
Quick Reference - ASL Alphabets


Real Time Communication System
127.0.0.1:5000

Real-Time Communication System Powered By AI For Specially Abled

The Predicted Alphabet is: H

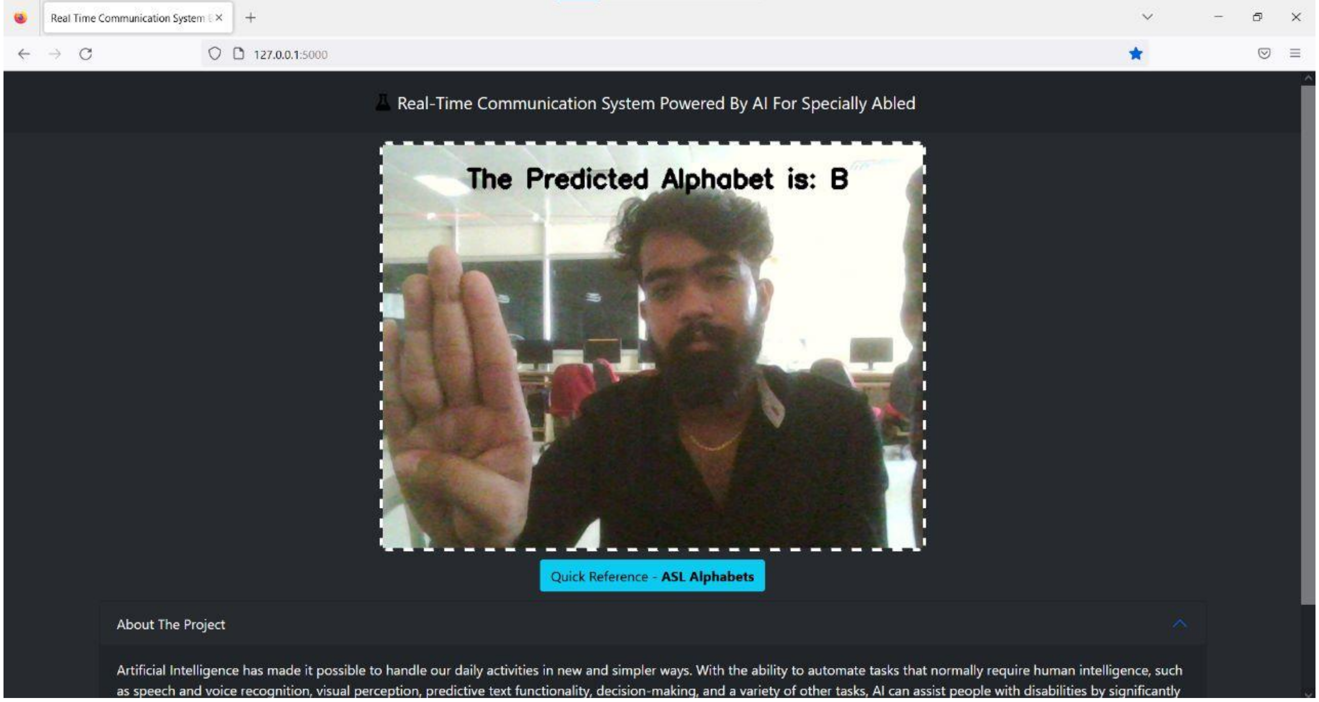
Quick Reference - ASL Alphabets
About The Project



Real Time Communication System
127.0.0.1:5000
Quick Reference - **ASL Alphabets**
About The Project

Real-Time Communication System Powered By AI For Specially Abled

The Predicted Alphabet is: D

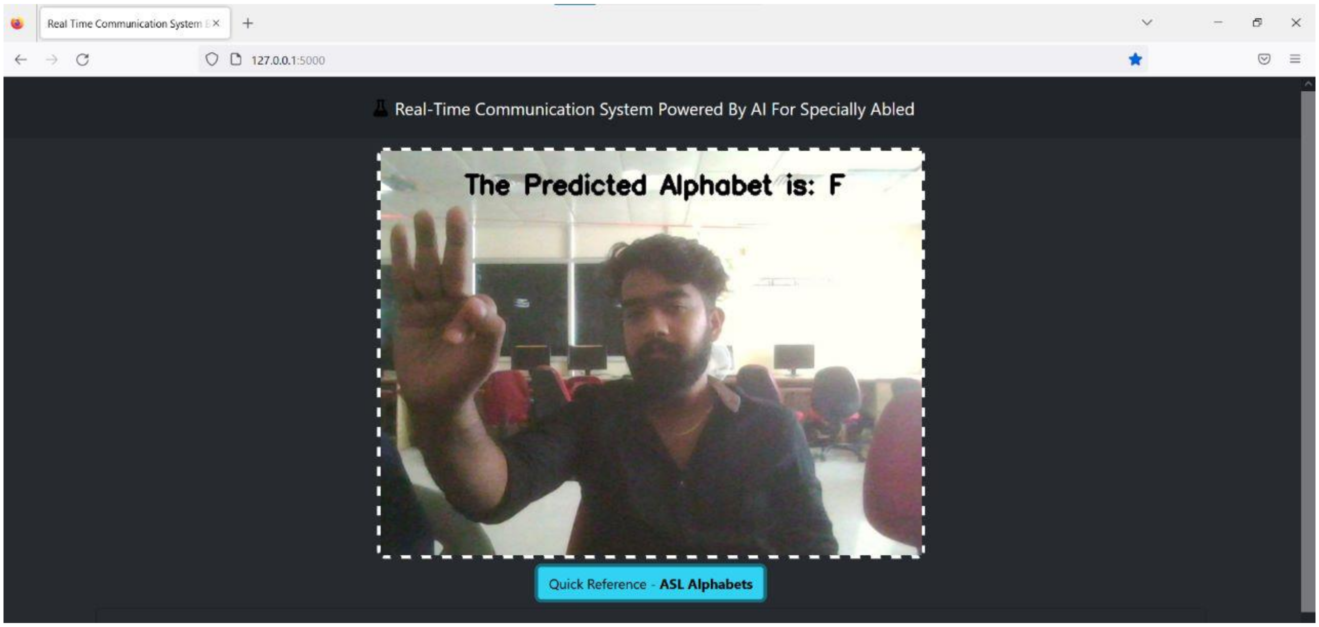


Real Time Communication System
127.0.0.1:5000
Quick Reference - **ASL Alphabets**
About The Project

Real-Time Communication System Powered By AI For Specially Abled

The Predicted Alphabet is: B

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


Real Time Communication System E
127.0.0.1:5000
Quick Reference - **ASL Alphabets**

Real-Time Communication System Powered By AI For Specially Abled

The Predicted Alphabet is: F

7. ADVANTAGES & DISADVANTAGES

Advantages:

1. It is possible to create a mobile application to bridge the communication gap betweendeaf and dumb persons and the general public.

2. As different sign language standards exist, their dataset can be added, and the user canchoose which sign language to read.

Disadvantages:

1. The current model only works from alphabets A to I.

2. In absence of gesture recognition, alphabets from J cannot be identified as they requiresome kind of gesture input from the user.

3. As the quantity/quality of images in the dataset is low, the accuracy is not great, but thatcan easily be improved by change in dataset.

8. APPLICATIONS

1. It will contribute to the development of improved communication for the deafened. Themajority of people are unable to communicate via sign language, which creates a barrierto communication
2. As a result, others will be able to learn and comprehend sign language and communicatewith the deaf and dumb via the web app.
3. According to scientific research, learning sign language improves cognitive abilities,attention span, and creativity.

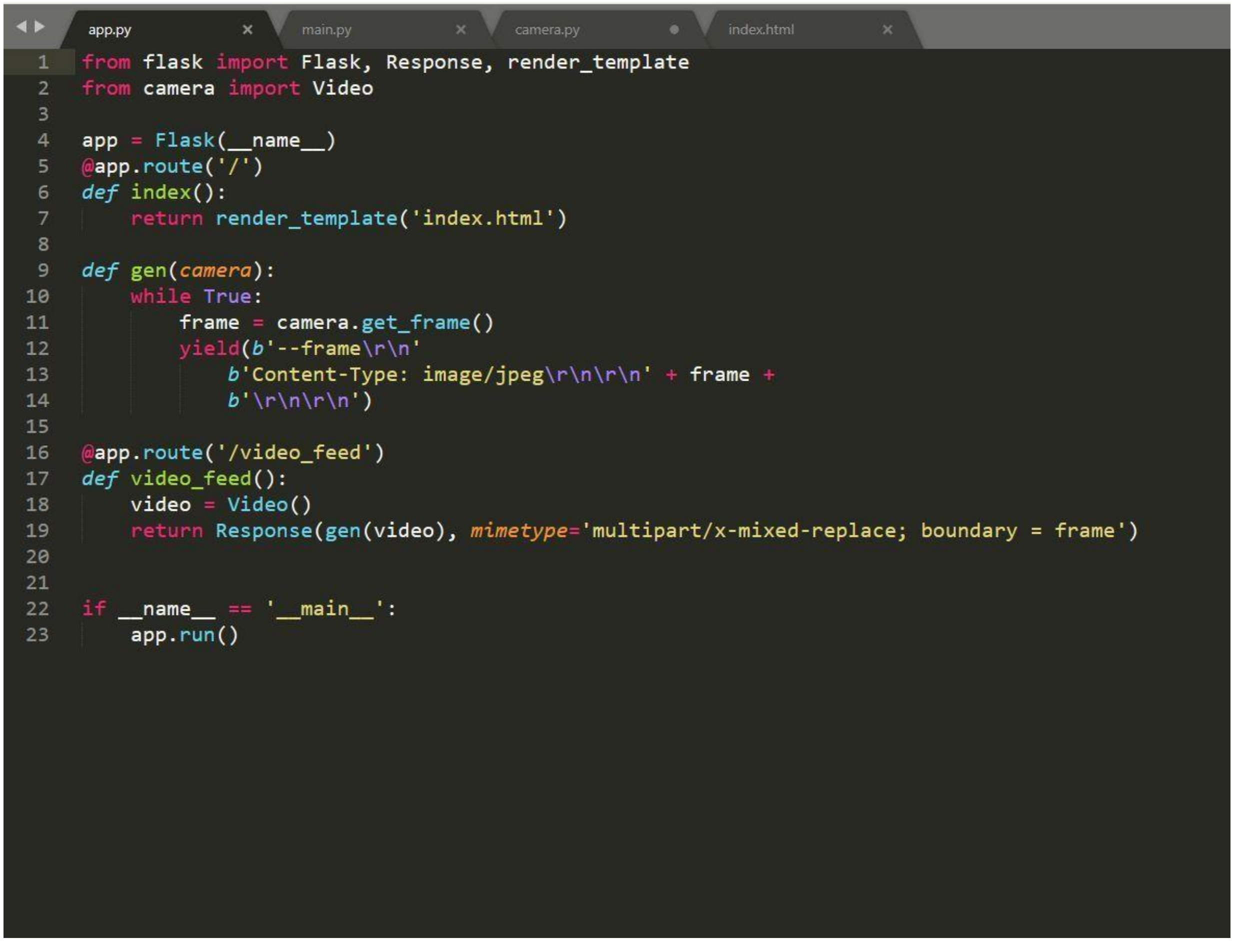
9. CONCLUSION

Sign language is a useful tool for facilitating communication between deaf and hearing people. Because it allows for two-way communication, the system aims to bridge the communication gapbetween deaf people and the rest of society. The proposed methodology translates language into English alphabets that are understandable to humans.

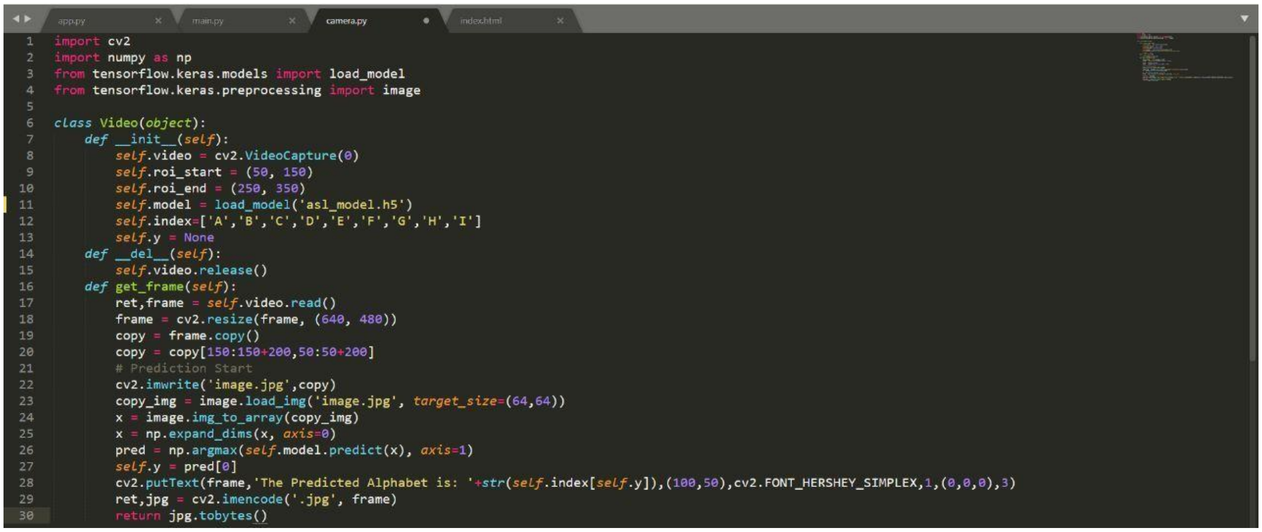
This system sends hand gestures to the model, who recognises them and displays the equivalentAlphabet on the screen. Deaf-mute people can use their hands to perform sign language, which will then be converted into alphabets, thanks to this project.

10. APPENDIX

Web app code



```python
from flask import Flask, Response, render_template
from camera import Video

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

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield(b'--frame\r\n'
            b'Content-Type: image/jpeg\r\n\r\n' + frame +
            b'\r\n\r\n')

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


if __name__ == '__main__':
    app.run()
```



```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')
        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()
```

American Sign Language Standard Reference: