

LOYOLA INSTITUTE OF TECHNOLOGY

REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED

TEAM ID: PNT2022TMID25606

MENTOR NAME: VARADHARAJAN M

INDUSTRY MENTOR: DIVYA

TEAM LEADER AISHWARYA R-210919106006

TEAM MEMBER: KANIMOZHI V-210919106037

TEAM MEMBER: JAYANTHINI S-210919106034

TEAM MEMBER: DHARANI M-210919106022

Project Report Format

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

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

3. IDEATION & PROPOSED SOLUTION

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

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

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

6. PROJECT PLANNING & SCHEDULING

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

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

- 7.1 Feature 1
- 7.2 Feature 2
- 7.3 Database Schema (if Applicable)

8. TESTING

- 8.1 Test Cases
- 8.2 User Acceptance Testing

9. RESULTS

- 9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

- Source Code
- GitHub & Project Demo Link

1. INTRODUCTION

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 timeshifting 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.

Purpose

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 is given as output.

Problem Statement Definition

Communication is the only medium by which we can share our thoughts or convey the message but 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

problem:

Vedha has difficulty in hearing. He uses sign language to communicate with others. But he can't able to communicate with normal people who don't understand sign language.

Solution:

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 the system enhances the user friendly experience.

problem:

Itam is a dumb by birth. He uses sign language to communicate with others. But he can't able to communicate with normal people who don't understand sign language.

solution:

To create people app for understanding sign language and convert into Speech signal as output for normal .

2. LITERATURE SURVEY

Existing Problem

In our society, we have people with disabilities. The technology is developing day by day but no significant developments are undertaken for the betterment of these people. Communication 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.

References

1. Koufos, K., EL Halou, 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.
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. Spec*, 1, 1-6.

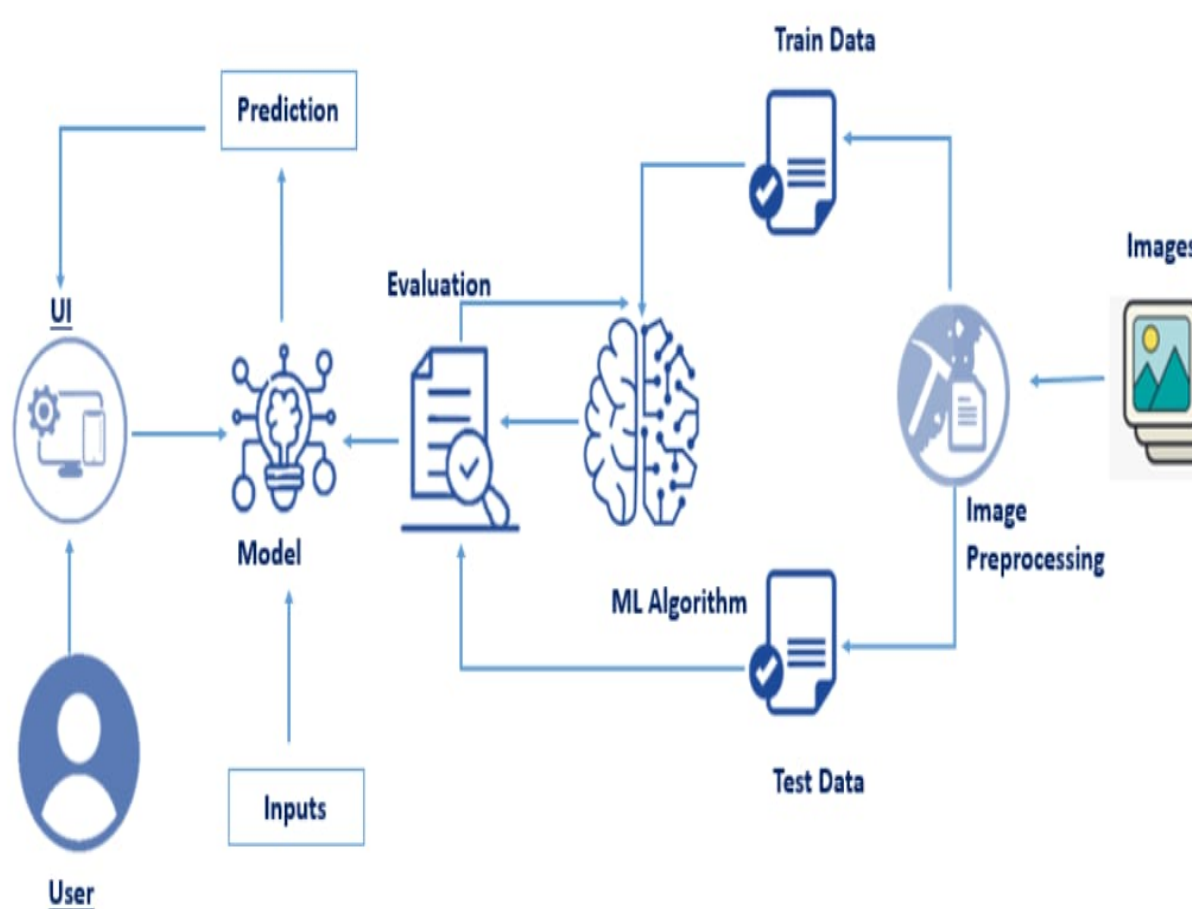
Problem Statement Definition

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 is given as output.

PROPOSED SOLUTION

S . N O	PARAM ETER	DESCRIPTION
1	PROBL EM STATE MENT	In our society, we have people with disabilities.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
2	SOLUTI ON DESCRI PTION	The human hand has remained a popular choice to convey information in situations where other forms like speech cannot be used. REAL TIME COMMUNICATION SYSTEM POWERED BY AL FOR SPECIALLY ABLED will be very useful to have a proper conversation between a normal person and an impaired person in any language

3	NOVELTY/UNIQUENESS	This app converts the sign language into a human hearing voice in the desired language to convey a message to normal people, as well as converts speech into understandable sign language for the deaf-mute
4	SOCIAL IMPACT	People with disabilities can drastically improve their everyday lives.
5	BUSINESS MODEL (REVENUE MODEL)	 <pre> graph LR User((User)) --> UI((UI)) UI --> Model((Model)) Inputs[Inputs] --> Model Model --> Prediction[Prediction] Prediction --> UI Model --> Evaluation[Evaluation] Evaluation --> ML[ML Algorithm] ML --> Evaluation ML --> Train[Train Data] ML --> Test[Test Data] Images[Images] --> Preprocessing((Image Preprocessing)) Preprocessing --> ML Preprocessing --> Train Preprocessing --> Test </pre> <p>The diagram illustrates the business model and technical workflow of the app. It shows a cycle between a User, UI, Model, Prediction, Evaluation, ML Algorithm, Image Preprocessing, and Images, with data flows for Train Data and Test Data.</p>

EMPATHY MAP

Team ID: PNT2022TMID25606
Project name: REAL TIME
COMMUNICATION SYSTEM
POWERED BY AI FOR
SPECIALLY ABLED
Maximum mark: 4 MARKS



REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENTS

S.NO	FUNCTIONAL REQUIREMENTS	SUB REQUIRMENTS
FR-1	User Registration	Registration through gmail or registration through moblie number
FR-2	User confirmation	Confirmation via Email or Confirmation via OTP
FR-3	System Requirements	1.moblie or PC or Laptop with webcam or camera 2.Minimum 1GB RAMand picture capability
FR-4	Text conversion	Converts the sign language into a text using CNN model
FR-5	sentence translation	To creat sentences by recognizing the signs and pauses in the video stream
FR-6	Speech translation	TTS converts text into speech

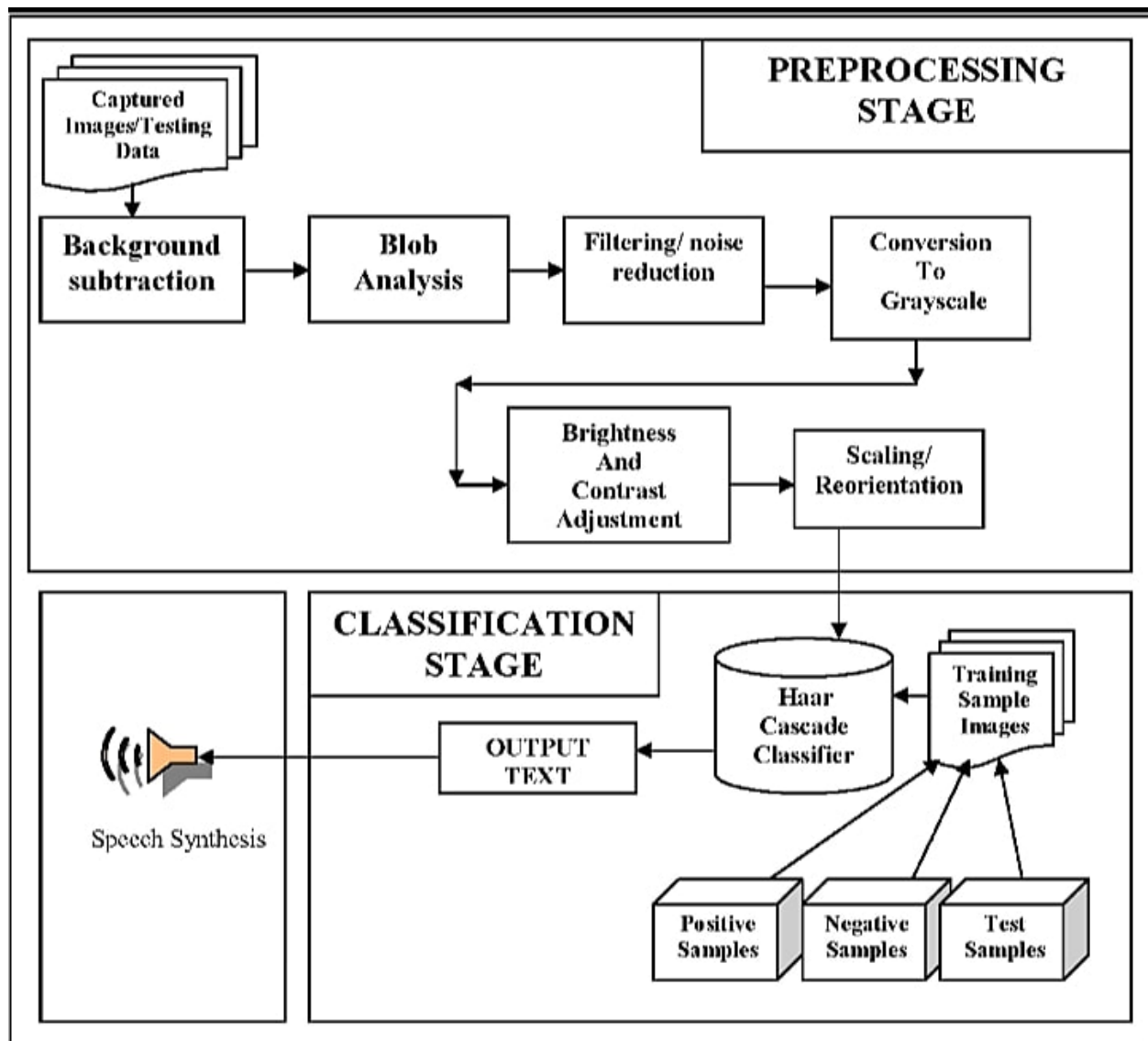
NON FUNCTIONAL REQUIREMENTS

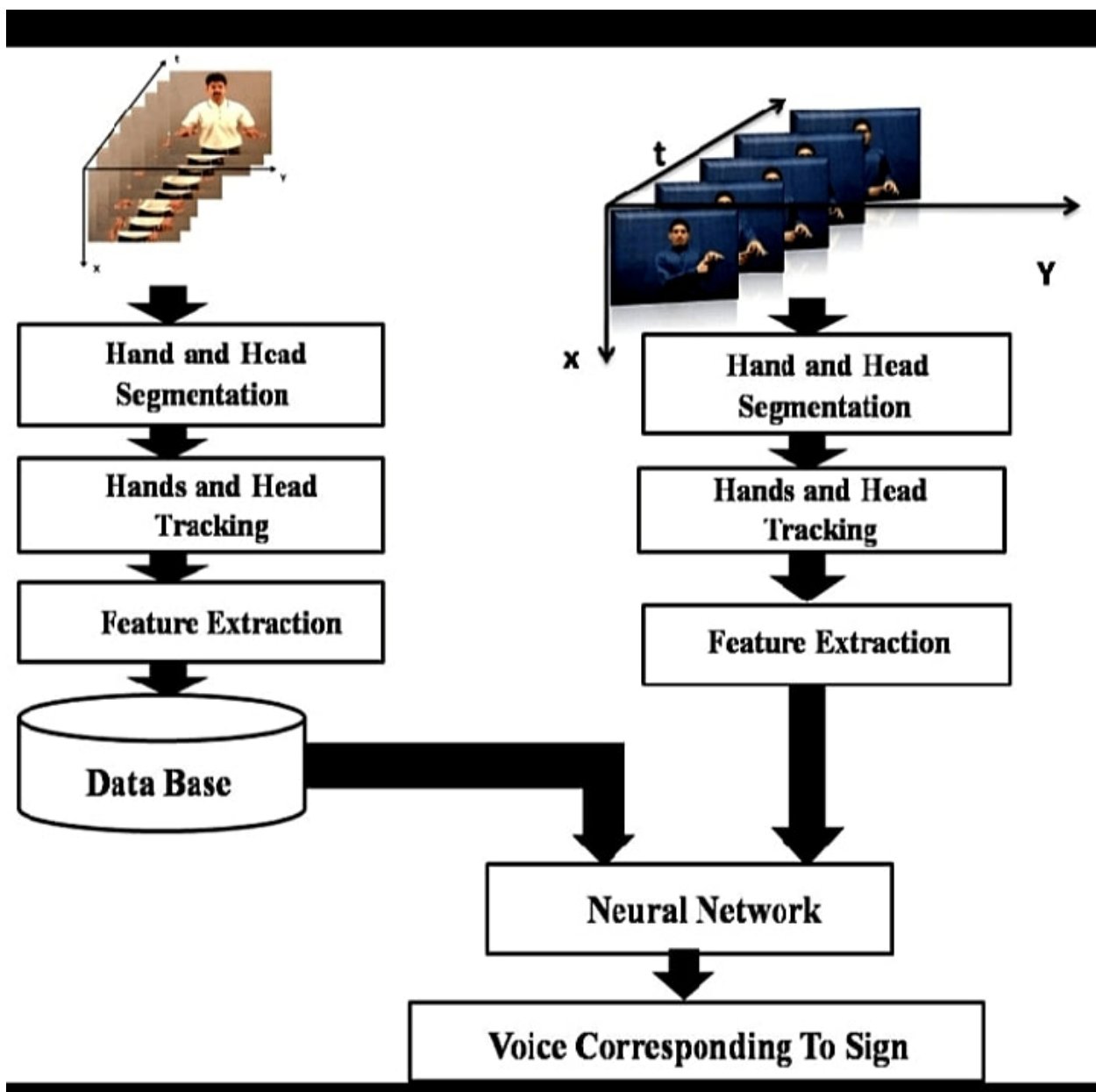
FR.NO	NON FUNCTIONAL REQUIREMENTS	DESCRIPTION
NFR1	Usability	Easy usable application for everyone.Especially useful for disabled person. It is user friendly
NFR-2	Security	It is also a secured application and information and images are securely stored.It must be ensured that the privacy of user data be maintained and handled appropriately.
NFR-3	Reliability	The translation of sign languages should be reliable. The accuracy of the system should be tested extensively to make sure that it is up to the mark.
NFR-4	Performance	It's performance is consistency good .The processing should be done in considerable time so that the conversation can go on without waiting for the system's output.

NFR-4	Availability	It is a free accessible and Universal access. Since sign language is almost the same everywhere, the system can be used across the globe

project design:

DATA FLOW DIAGRAM:

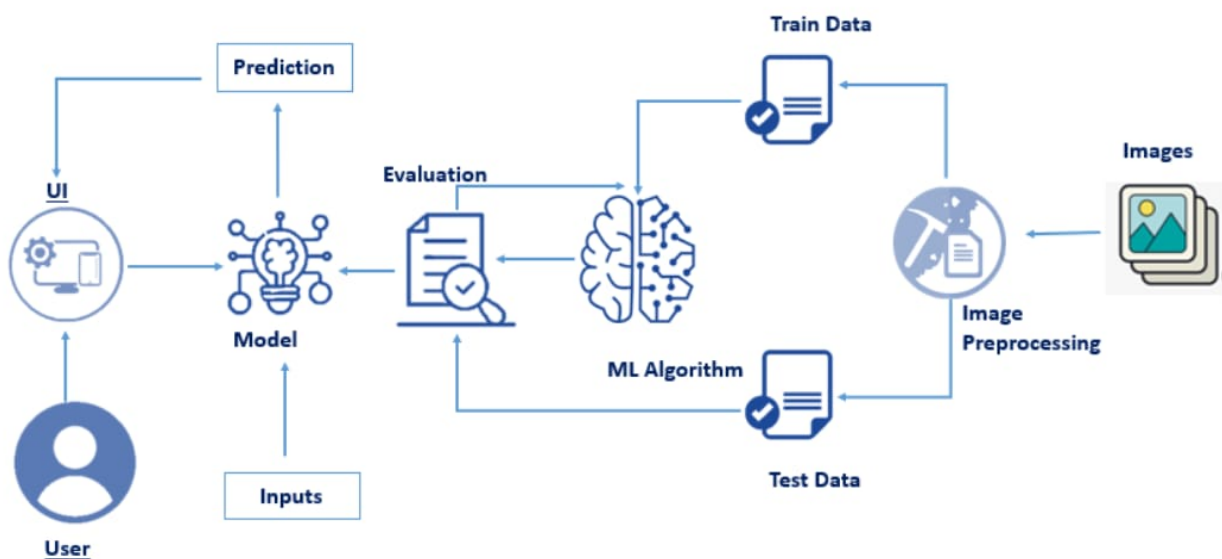


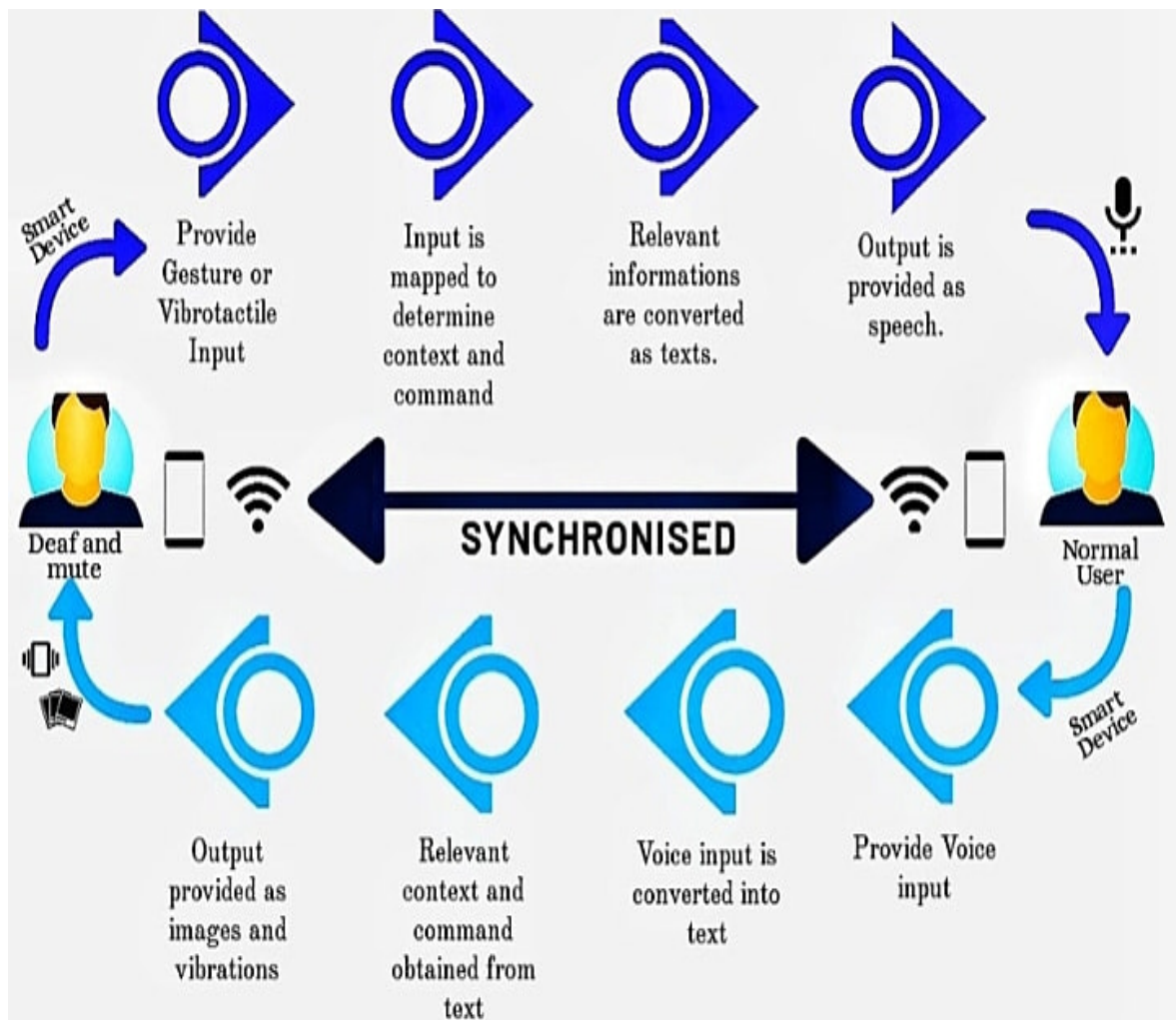


Solution Architecture:

Solution architecture is a complex process - with many sub-processes - that bridges the gap between business problems and technology solutions. Its goals are to:

- * Find the best tech solution to solve existing business problems.
- * Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- * Define features, development phases, and solution requirements.
- * Provide specifications according to which the solution is defined, managed, and delivered.





MILESTONE ACTIVITY PLAN

MILESTONE	FUNCTION	MILESTONE STORY NUMBER	STORY/TASK
MILESTONE-1	Data collection	M1	we're collecting dataset for building our project and creating two folders, one for training and another one for testing
MILESTONE-2	Image preprocessing	M2	Importing image data generator libraries and applying image data generator functionality to train the test set
MILESTONE-3	Model building	M3	importing the model building libraries, Initializing the model, Adding Convolution layers, Adding the Pooling

			layers, Adding the Flatten layers, Adding Dense layers, Compiling the model fit and save the model
MILESTONE-4	Testing model	M4	import the packages first. Then we save the model and Load the test image, preprocess it and predict it.
MILESTONE-5	Application layer	M5	Build the flask application and the HTML pages.
MILESTONE-6	Train CNN model	M6	Register for IBM Cloud and train ImageClassification Model
MILESTONE-7	Final result	M7	To ensure all the activities and resulting the final output.

SPRINT SCHEDULE:

SPRINT	FUNCTIONAL REQUIREMENTS	USER STORY NUMBER	USER STORY/TASK	STORY POINTS	PRIORITY	TEAM MEMBERS
SPRINT-1	Data collection	USN-1	collect dataset	9	high	R.AISHWARYA
SPRINT-1		USN-2	image processing	8	medium	S.JAYANTHINI V.KANIMOZHI
SPRINT-2	Model building	USN-3	import the required libraries, add the necessary layers and complete model	10	high	V.KANIMOZHI M.DHARANI
SPRINT-2		USN-4	Training the image classification model using CNN	7	medium	M.DHARANI
SPRINT-3	Training and testing	USN-5	Training the model and testing the models performance	9	high	S.JAYANTHINI

SPRINT-4	Implementation of the application	USN-6	Converting the input sign language images into English alphabets	8	medium	R.AISHWARYA
----------	-----------------------------------	-------	--	---	--------	-------------

PROJECT TRACKER:

SPRINT	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE(PLANNED)	STORY POINTS COMPLETED	SPRINT RELEASE DATE(ACTUAL)
SPRINT-1	10	6 days	24 OCT 2022	29 OCT 2022	8	29 OCT 2022
SPRINT-2	10	6 days	31 OCT 2022	04 NOV 2022	5	04 NOV 2022
SPRINT-3	10	6 days	07 NOV 2022	11 NOV 2022	7	11 NOV 2022
SPRINT-4	10	6 days	14 NOV 2022	18 NOV 2022	5	18 NOV 2022

VELOCITY:

AV =sprint duration/velocity

$$AV=6/10=0.6$$

BURNDOWN CHART

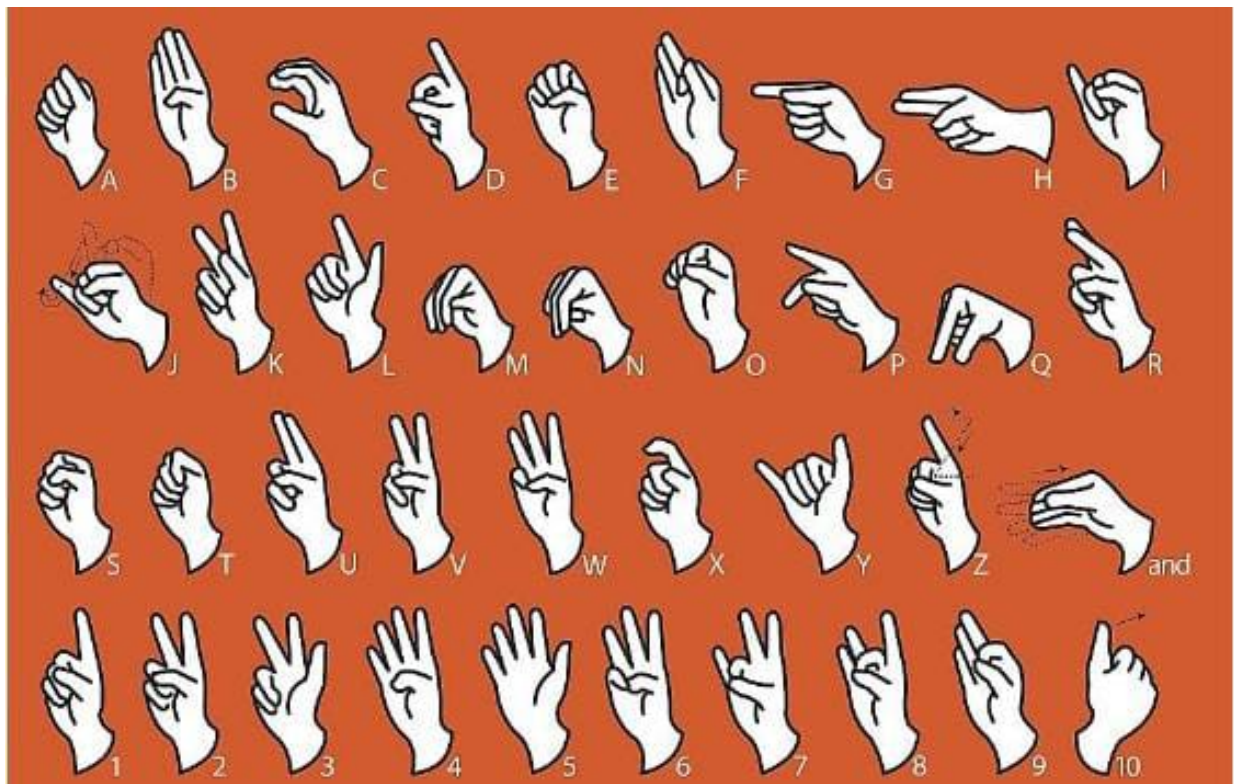


SPRINT BURNDOWN CHART



CODING AND SOLUTION

The user can choose which sign language to read based on the different sign language standards that exist.



MODEL BUILDING

```
from keras.models import Sequential
```

```
from keras.layers import Dense
```

```
from keras.layers import Convolution2D
```

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D
```

```
from keras.layers import Dropout
```

```
from keras.layers import Flatten
```

```
#Creating the model
```

In [101]:

```
model=Sequential()
```

```
#Adding the layers
```

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

```
model.add(MaxPooling2D(pool_size=(2,2))
```

```
model.add(Flatten()))
```

```
#adding hidden layers
```

```
model.add(Dense(400, activation='relu'))
```

```
model.add(Dense(200, activation='relu'))
```

```
model.add(Dense(100, activation='relu'))
```

```
#Adding the output layer
```

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

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

```
model.fit_generator(x_train, steps_per_epoch=30,  
epochs=10,validation_data=x_test,validation_steps=50)
```

Epoch 1/10

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning:

`Model.fit_generator` is deprecated and will be removed in a future version. Please use
`Model.fit`, which supports generators.

"""Entry point for launching an IPython kernel

30/30 [=====] - ETA: 0s - loss: 0.0083-
accuracy:0.9957

WARNING:tensorflow:Your input ran out of data; interrupting
training.Make sure that yourdataset or generator can generate at least
`steps_per_epoch * epochs` batches (in this case, 50batches). You may
need touse the repeat() function when building your dataset.

30/30 [=====] - 18s 587ms/step - loss: 0.0083 -
accuracy:0.9957 - val_loss: 0.2910 val_accuracy: 0.9693

Epoch 2/10

30/30 [=====] - 12s 402ms/step - loss: 0.0081 -
accuracy:0.9980

Epoch 3/10

30/30 [=====] - 12s 400ms/step - loss: 0.0102 -
accuracy:0.9963

Epoch 4/10

30/30 [=====] - 12s 402ms/step - loss: 0.0049 -
accuracy:0.9993

Epoch 5/10

30/30 [=====] - 12s 402ms/step - loss: 0.0030 -
accuracy:0.9997

Epoch 6/10

30/30 [=====] - 12s 394ms/step - loss: 0.0019 -
accuracy:0.9997

Epoch 7/10

30/30 [=====] - 12s 401ms/step - loss: 0.0081 -
accuracy:0.9973

Epoch 8/10

30/30 [=====] - 12s 402ms/step - loss: 0.0124 -
accuracy:0.9960

Epoch 9/10

30/30 [=====] - 12s 401ms/step - loss: 0.0070 -
accuracy:0.9987

Epoch 10/10

30/30 [=====] - 12s 399ms/step - loss: 0.0089 -
accuracy:0.9973

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

TEST THE MODEL

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

```
model = load_model('/content/Real_time.h5')
```

```
img = image.load_img('/content/Dataset/test_set/H/107.png',target_size (100,100))img
```



```
from skimage.transform import resize
```

```
def detect(frame):
```

```
    img=image.img_to_array(frame)
```

```
    img = resize(img,(64,64,1))
```

```
    img = np.expand_dims(img,axis=0)
```

```
    pred=np.argmax(model.predict(img))
```

```
    op=[' A','B','C','D','E','F','G','H','I']
```

```
    print("THE PREDICTED LETTER IS ",op[pred])
```

```
img=image.load_img("/content/Dataset/test_set/H/107.png")
```

```
detect(img)
```

```
1/1 [=====] - 0s 28ms/step
```

```
THE PREDICTED LETTER IS H
```

```
img = image.load_img('/content/Dataset/test_set/A/110.png')
```

```
pred=detect(img)
```

```
1/1 [=====] - 0s 26ms/step
```

```
THE PREDICTED LETTER IS A
```

```
img=image.load_img('/content/Dataset/test_set/E/111.png')
```

```
detect(img)
```

```
1/1 [=====] - 0s 30ms/step
```

```
THE PREDICTED LETTER IS E
```


FEATURE 2

The communication gap between deaf and dumb people and the general public can be bridged with a mobile application.

Mobile App:

```
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')

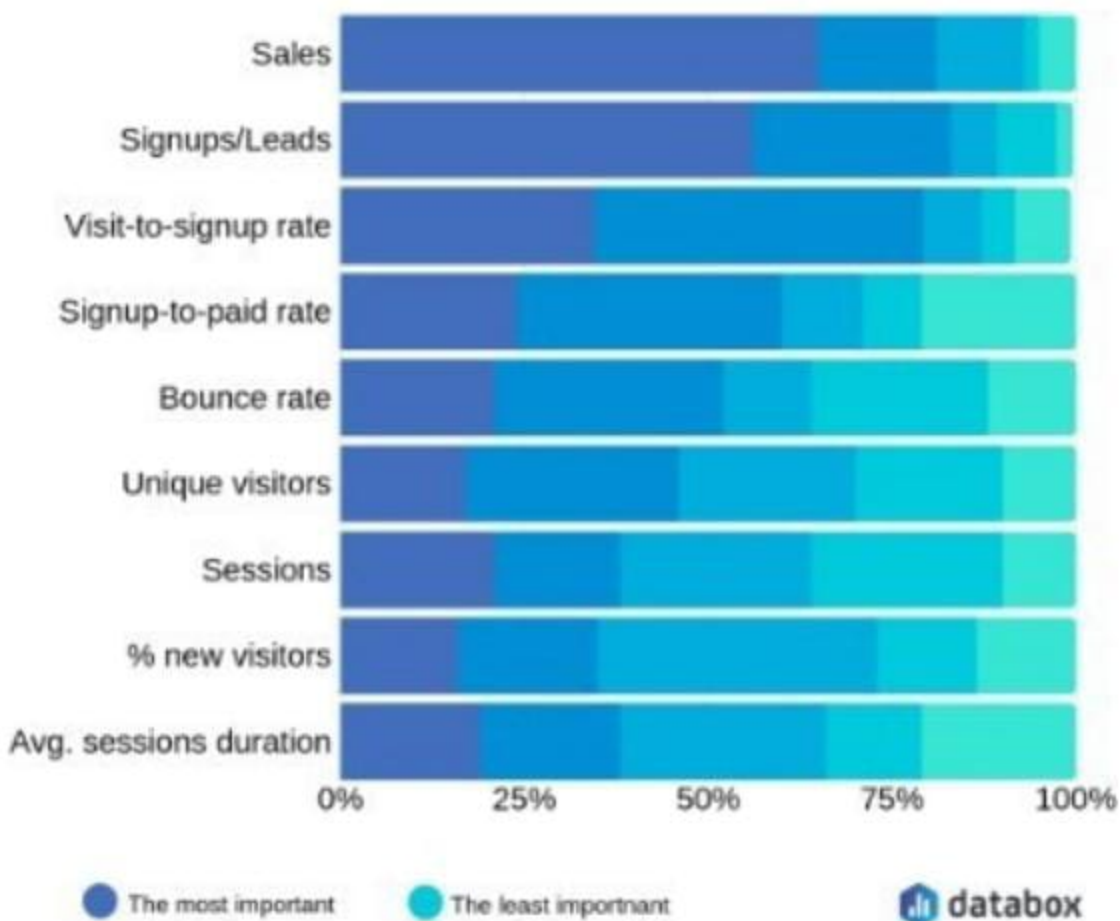
if __name__ == '__main__':
    app.run()
```

RESULTS

The proposed procedure was implemented and tested on a set of images.

The training database consists of 15750 images of Alphabets from "A" to "I", while the testing database consists of 225 images of Alphabets from "A" to "I"

Once the gesture is recognized the equivalent alphabet is shown on the screen.



OUTPUT



ADVANTAGES AND DISADVANTAGES

Advantages:

The speech is converted to sign language very quick to provide greater and faster understanding to specially-abled people.

The user interface is convenient and simple for both people.

Disadvantages:

The number of images and pixels for the model to train in the dataset is not high so accuracy is moderate level.

It will be improved by changing the dataset.

Currently, we have deployed a dataset in the model for the alphabets A to I only.

CONCLUSION:

It aims to bridge the communication gap between deaf mute people and the rest of society. The proposed methodology translates sign language into English alphabets that are understandable to humans. This system sends hand gestures to the model, who recognizes them and displays the equivalent.

FUTURE SCOPE:

With the introduction of gesture recognition, the web app can easily be expanded to recognize letters beyond 'I', digits, and other symbols plus gesture recognition can also allow controlling of software/hardware interfaces. Having a technology that can translate hand sign language to its corresponding alphabet is a game changer in the field of communication and AI for specially-abled people such as those deaf or

dumb.

IMAGE PREPROCESSING:

image_dataset_from_directory

```
tf.keras.preprocessing.image_dataset_from_directory(  
    directory,  
    labels="inferred",  
    label_mode="int",  
    class_names=None,  
    color_mode="rgb",  
    batch_size=32,  
    image_size=(256, 256),  
    shuffle=True,  
    seed=None,  
    validation_split=None,  
    subset=None,  
    interpolation="bilinear",  
    follow_links=False,  
    crop_to_aspect_ratio=False,  
    **kwargs  
)
```

GENERATES A `tf.data.Dataset` from image files in a directory

```
main_directory/  
...class_a/  
.....a_image_1.jpg  
.....a_image_2.jpg  
...class_b/  
.....b_image_1.jpg  
.....b_image_2.jpg
```

load_image function

```
tf.keras.preprocessing.image.load_img(  
    path, grayscale=False, color_mode="rgb", target_size=None, interpolation="nearest"  
)
```

Loads an image into PIL FORMATE

```
image = tf.keras.preprocessing.image.load_img(image_path)
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.array([input_arr]) # Convert single image to a batch.
predictions = model.predict(input_arr)
```

image_to_array funtion

```
tf.keras.preprocessing.image.img_to_array(img,data_format=None, dtype=None)
```

Converts a PIL image instance to a numpy array

```
from PIL import Image
img_data = np.random.random(size=(100, 100, 3))
img = tf.keras.preprocessing.image.array_to_img(img_data)
array = tf.keras.preprocessing.image.img_to_array(img)
```

MODEL BULIDIND:

Initialize The Mode:

Initialize the neural network layer by creating a reference/object to the Sequential class.

```
model=Sequential()
```

Add The Convolution Layer

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution2D class is used. It takes the number of feature detectors, feature detector size, expected input shape of the image, activation function as arguments. This layer applies feature detectors on the input image and returns a feature map (features from the image).

```
model.add(Convolution2D(32, (3,3), input_shape=(64,64,1), activation = 'relu'))  
#no. of feature detectors, size of featurdetector, image size, activation function
```

Add The Pooling Layer

After the convolution layer, usually, the pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. The efficient size of the pooling matrix is (2,2). It returns the pooled feature maps. (Note: Any number of convolution layers, pooling and dropout layers can be added)

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

Add the flatten layer:

The flatten layer is used to convert the n-dimensional array to a 1-dimensional array. This 1D array will be given as input to ANN layers.

```
model.add(Flatten())
```

Adding The Dense Layers

Three dense layers are added which usually takes the number of units/neurons. Specifying the activation function, kind of weight initialization is optional.

```
model.add(Dense(units=512, activation='relu'))
```

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

Compile The Model

After adding all the required layers, the model is to be compiled. For this step, loss function, optimizer, and metrics for evaluation can be passed as arguments

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

Fit And Save The Model

Fit the neural network model with the train and test set, number of epochs, and validation steps.

```
model.fit_generator(x_train, steps_per_epoch=24, epochs=10, validation_data=x_test, validation_steps=40)  
#steps_per_epoch = no. of train images//batch size
```

The weights are to be saved for future use. The weights are saved in as .h5 file using save().

```
model.save('aslpng1.h5')
```

TEST THE MODEL

Import The Packages And Load The Saved Model

As a first step to start prediction we import packages that are used for loading the model and used to expand the dimension of the image. We use the Keras package to load the model which was saved when we built the model.

```
from keras.models import load_model  
import numpy as np  
import cv2
```



```
model=load_model('aslpng1.h5')
```

Load The Test Image, Pre-Process It And Predict

Pre-processing the image includes converting the image to the array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.

```
from skimage.transform import resize
def detect(frame):
    img = resize(frame,(64,64,1))
    img = np.expand_dims(img,axis=0)
    if(np.max(img)>1):
        img = img/255.0
    prediction = model.predict(img)
    print(prediction)
    prediction = model.predict_classes(img)
    print(prediction)

frame=cv2.imread(r"G:\Gayatri Files\Smartbridge\Widhi\Conversation Engine for Deaf and Dumb\Dataset\test_set\G\1.png")
data = detect(frame)

[[6.0201724e-13  7.6744452e-18  1.7007801e-10  7.7269103e-14  2.9694178e-15
  8.9405344e-16  9.9999082e-01  9.1214142e-06  3.0555274e-17]]
[6]
```

APPLICATION BUILDING

BUILDING CAMERA:

```
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):
```

```
    k = cv2.waitKey(1)
```

```
    self.video.release()
```

```
def get_frame(self):
```

```
    ret, frame = self.video.read()
```

```
    frame = cv2.resize(frame, (64...
```

```
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()
```

```
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'
```

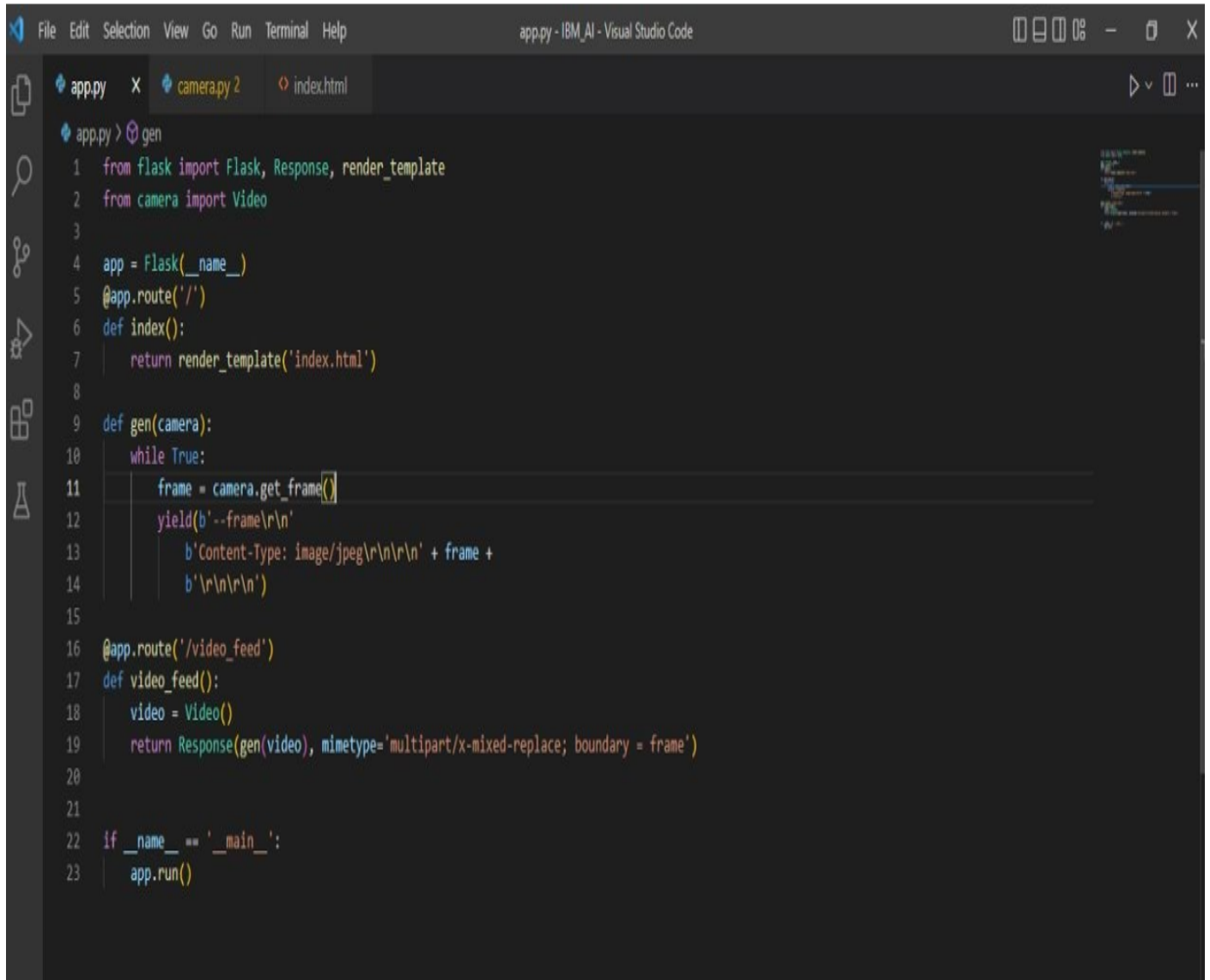
```
              b'content-Type: image/jpeg\r\n\r\n'+ frame +
```

```
        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()
```

APPENDIC

SOURCE CODE:

FLASK:



The image shows a Visual Studio Code editor window with the title bar "app.py - IBM_AI - Visual Studio Code". The editor has three tabs open: "app.py", "camera.py 2", and "index.html". The "app.py" tab is active, displaying the following Python code:

```
1 from flask import Flask, Response, render_template
2 from camera import Video
3
4 app = Flask(__name__)
5 @app.route('/')
6 def index():
7     return render_template('index.html')
8
9 def gen(camera):
10     while True:
11         frame = camera.get_frame()
12         yield(b'--frame\r\n'
13              b'Content-Type: image/jpeg\r\n\r\n' + frame +
14              b'\r\n\r\n')
15
16 @app.route('/video_feed')
17 def video_feed():
18     video = Video()
19     return Response(gen(video), mimetype='multipart/x-mixed-replace; boundary = frame')
20
21
22 if __name__ == '__main__':
23     app.run()
```

HTML:

```
File Edit Selection View Go Run Terminal Help • index.html - IBM_AI - Visual Studio Code
template > index.html > html > head
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
7   <title>REAL TIME COMMUNICATION </title>
8   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
9   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
10  <link rel="stylesheet" href="Navbar-Centered-Brand.css">
11 </head>
12
13 <body style="background: #f5ad41;">
14   <nav class="navbar navbar-light navbar-expand-md py-3" style="background: #22697a;">
15     <div class="container">
16       <div></div><div><a class="navbar-brand d-flex align-items-center" href="#"><span
17         class="bs-icon-sm bs-icon-rounded bs-icon-primary d-flex justify-content-center align-items-center me-2 bs-icon"><i
18         class="fas fa-flask"></i></span><h4 style="color: #a5eb24; font-style: oblique; text-align: center;"><strong> Real-Time
19         System Powered By AI&nbsp;For Specially Abled</strong></h4></a>
20       </div></div>
21     </nav>
22   <div>
23     <h2 style="text-align: center; webkit-text-fill-color: #045816;"><strong>TEAMID-- PNT2022TMID40538</strong></h2>
24   </div>
25   <section>
26     <div class="d-flex flex-column justify-content-center align-items-center">
27       <div class="d-flex flex-column justify-content-center align-items-center" id="div-video-feed"
28         style="width: 800px; height: 600px; margin: 10px; min-height: 480px; min-width: 640px; border-radius: 50px; border: 10px groove #0458
29         
31       </div>
32     </div>
33   </section>
```

```
template > index.html > html > body > section > div.container > div.accordion-1.accordion.text-white > div.accordion-item > div.accordion-collapse.collapse.item-2 > div.accordion-body
33 </div>
34 <div class="d-flex flex-column justify-content-center align-items-center" style="margin-bottom: 20px;"><button
35   class="btn btn-info" type="button" data-bs-target="#modal-1" data-bs-toggle="modal">Quick Reference
36   <strong> ASL Alphabets</strong></button></div>
37 </section>
38 <section>
39   <div class="container">
40     <div class="accordion text-white" role="tablist" id="accordion-1">
41       <div class="accordion-item" style="font-style: oblique; background: rgb(33,37,41);">
42         <h2 class="accordion-header" role="tab"><button class="accordion-button" data-bs-toggle="collapse"
43           data-bs-target="#accordion-1 .item-1" aria-expanded="true"
44           aria-controls="accordion-1 .item-1"
45           style="font-style: inherit; background: #3e6d9c; color: rgb(255,255,255);">About The Project</button></h2>
46         <div class="accordion-collapse collapse show item-1" role="tabpanel" data-bs-parent="#accordion-1">
47           <div class="accordion-body">
48             <p class="mb-0">In our society, we have people with disabilities. The technology is developing day by day but no sign
49           </div>
50         </div>
51       </div>
52       <div class="accordion-item" style="font-style: oblique; background: rgb(33,37,41);">
53         <h2 class="accordion-header" role="tab"><button class="accordion-button collapsed"
54           data-bs-toggle="collapse" data-bs-target="#accordion-1 .item-2" aria-expanded="false"
55           aria-controls="accordion-1 .item-2"
56           style="font-style: oblique; background: #3e6d9c; color: rgb(231,241,255);">Developed By</button></h2>
57         <div class="accordion-collapse collapse item-2" role="tabpanel" data-bs-parent="#accordion-1">
58           <div class="accordion-body">
59             <p class="mb-0">Students From ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY<br><br>TEAM ID-- <strong>PNT2022TMID40
60             <strong>HARIPRASAD J</strong> 513519106006<br>3. <strong>PAVANKUMAR M</strong> 513519106014<br>4. <strong>YUVARAJ
61           </p>
62         </div>
63       </div>
64     </div>
```

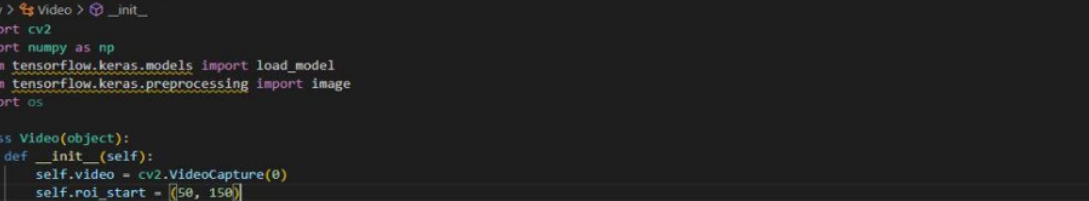
```

File Edit Selection View Go Run Terminal Help
index.html - IBM AI - Visual Studio Code

app.py camera.py 2 index.html
template > index.html > html
54 <h2 class="accordion-header" role="tab" >button class="accordion-button collapsed
55 data-bs-toggle="collapse" data-bs-target="#accordion-1 .item-2" aria-expanded="false"
56 aria-controls="accordion-1 .item-2"
57 style="font-style: oblique; background: linear-gradient(to right, #3E609C, #000000); color: white; padding: 5px 10px;">Developed By</button></h2>
58 <div class="accordion-collapse collapse item-2" role="tabpanel" data-bs-parent="#accordion-1">
59 <div class="accordion-body">
60 <p class="mb-0">Students From ANNAM MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY<br><br>TEAM ID-- <strong>PNT2022TMID40
61 <strong>HARIPRASAD J</strong> 513519106006<br>3. <strong>PAVANKUMAR M</strong> 513519106014<br>4. <strong>YUVARAJ
62 </p>
63 </div>
64 </div>
65 </div>
66 </div>
67 </section>
68 <div class="modal fade" role="dialog" tabindex="-1" id="modal-1">
69 <div class="modal-dialog" role="document">
70 <div class="modal-content">
71 <div class="modal-header">
72 <h4 class="modal-title">American Sign Language - Alphabets</h4><button type="button"
73 class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
74 </div>
75 <div class="modal-body"></div>
76 <div class="modal-footer"><button class="btn btn-secondary" type="button"
77 data-bs-dismiss="modal">Close</button></div>
78 </div>
79 </div>
80 </div>
81 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
82 </body>
83
84 </html>

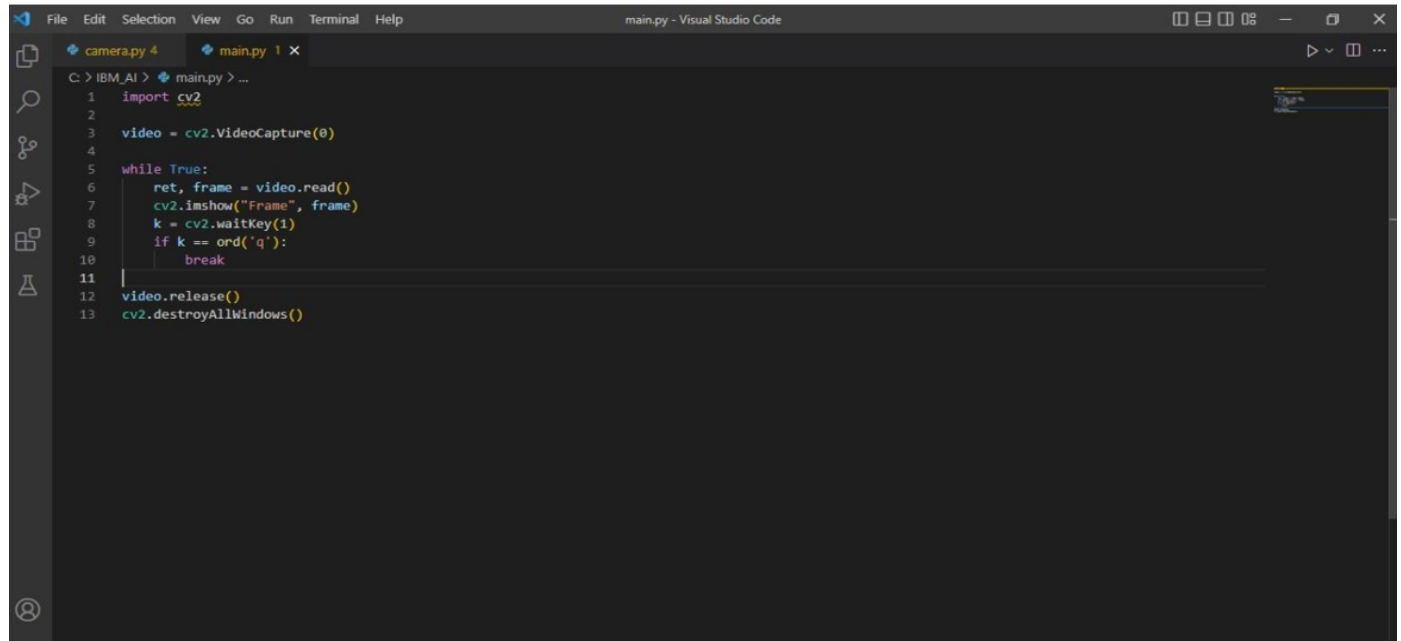
```

CAMERA:



```
1 import cv2
2 import numpy as np
3 from tensorflow.keras.models import load_model
4 from tensorflow.keras.preprocessing import image
5 import os
6
7 class Video(object):
8     def __init__(self):
9         self.video = cv2.VideoCapture(0)
10        self.roi_start = ([50, 150])
11        self.roi_end = (250, 350)
12        #self.model = load_model('asl_model.h5') # Execute Local Trained Model
13        self.model = load_model('asl_model.h5') # Execute IBM Trained Model
14        self.index=['A','B','C','D','E','F','G','H','I']
15        self.y = None
16    def __del__(self):
17        k = cv2.waitKey(1)
18
19        self.video.release()
20    def get_frame(self):
21        ret, frame = self.video.read()
22        frame = cv2.resize(frame, (640,480))
23        copy = frame.copy()
24        copy = copy[150:150+200,50:50+200]
25        # prediction starts
26        cv2.imwrite('image.jpg',copy)
27        copy_img = image.load_img('image.jpg', target_size=(64,64,3))
28        x = image.img_to_array(copy_img)
29        x = np.expand_dims(x, axis=0)
30        pred = np.argmax(self.model.predict(x), axis=1)
31        self.y = pred[0]
32        cv2.putText(frame, 'The Predicted Alphabet is: '+str(self.index[self.y]), (100,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 3)
```

MAIN:



```
File Edit Selection View Go Run Terminal Help
main.py - Visual Studio Code

camera.py 4 main.py 1 x

C:\> IBM_AI > main.py > ...
1 import cv2
2
3 video = cv2.VideoCapture(0)
4
5 while True:
6     ret, frame = video.read()
7     cv2.imshow("Frame", frame)
8     k = cv2.waitKey(1)
9     if k == ord('q'):
10         break
11
12 video.release()
13 cv2.destroyAllWindows()
```

TRAINED MODEL:

```
File Edit Selection View Go Run Terminal Help
Model_Training_Communication_Project.ipynb - Visual Studio Code

Model_Training_Communication_Project.ipynb
x New File Communication_Specialty_Hello_World_1 Project 001 Training_001 Model_Training_Communication_Project.ipynb 1 #Model Training for Real Time Communication through AI for Specialty Abled

Code Markdown Run All Clear Outputs of All Cells Outline
8
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Training Datasets
train_datagen = ImageDataGenerator(rescale=1/255, zoom_range=0.2, horizontal_flip=True, vertical_flip=False)
test_datagen = ImageDataGenerator(rescale=1/255)

# Training Dataset
train=train_datagen.flow_from_directory(r'F:\Projects\SmartBridge\Hussein\dataset\training_set', target_size=(64, 64), class_mode='categorical', batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'F:\Projects\SmartBridge\Hussein\dataset\test_set', target_size=(64, 64), class_mode='categorical', batch_size=900)

Found 27986 images belonging to 9 classes.
Found 25727 images belonging to 9 classes.

print("len x-train : ", len(x_train))
print("len x-test : ", len(x_test))

len x-train : 36
len x-test : 29

# The Class Indices in Training Dataset
train.class_indices

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

Model Creation

# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten, Dense
```