LOYOLA INSTITUTE OF TECHNOLOGY

REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED

TEAM ID: PNT2022TMID25606

MENTOR NAME : VARADHARAJAN M

TEAM LEADER : AISHWARYA R – 210919106006
TEAM MEMBER: KANIMOZHI V – 210919106037
TEAM MEMBER: JAYANTHINI S- 210919106034
TEAM MEMBER: DHARANI M - 210919106022

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

Real-time communication (RTC) refers to any communication that happens between two (or more) individuals in real-time – with minimal latency and without transmission delays. Some examples of real-time communication include landline phones, mobile calls, instant messaging, VoIP, and video conferencing.

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 onhand 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 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.

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 humanunderstandable language and speech is given as output.

IDEATION AND PROPOSED SOLUTION

EMPATHY MAP CANVAS

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

IDEATION AND BRAINSTORMING

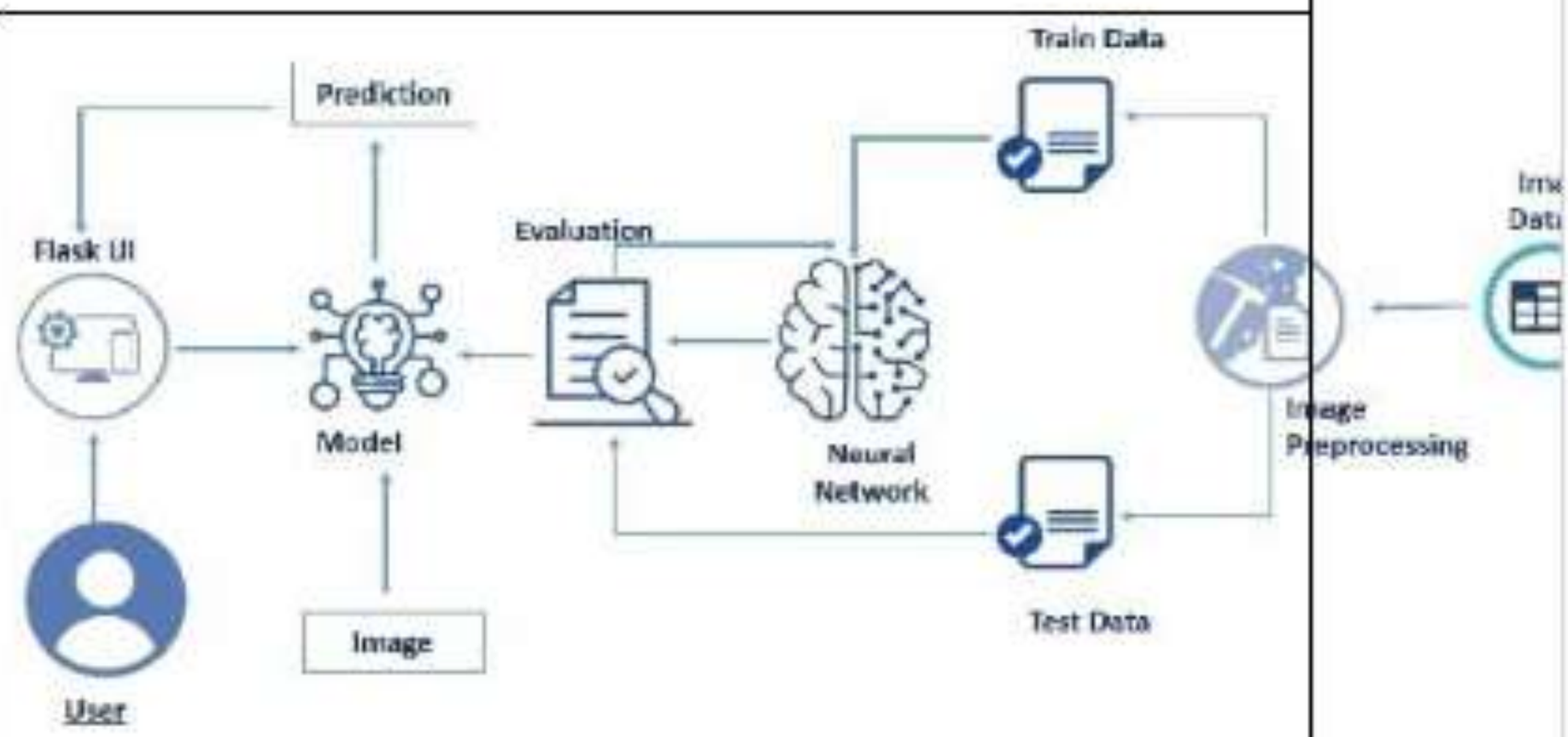
TEAM ID:PNT2022TMID25606

Project Design Phase-I

Proposed Solution Template

Team ID	PNT2022TMID25604
Project Name	REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED
Maximum Marks	2 Marks

Proposed Solution Template:

S.No	Parameter	Description
1.	Problem Statement(Problem to be solved)	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.	Solution description	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 AI 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)	
6.	Scalability of the	✓ adoption of mobile devices into consumers daily lives. ✓ It helps to understand verbal communication easily.

Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	03 October 2022
Team ID	PNT2022TMID25606
Project Name	Real time communication system powered by AI for specially-Abled
Maximum Marks	4 Marks

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 Gmail or Registration through Mobile number
FR-2	User Confirmation	Confirmation via Email or Confirmation via OTP
FR-3	System Requirements	1.Mobile or PC or Laptop with webcam or camera 2.Minimum 1GB RAM and 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.

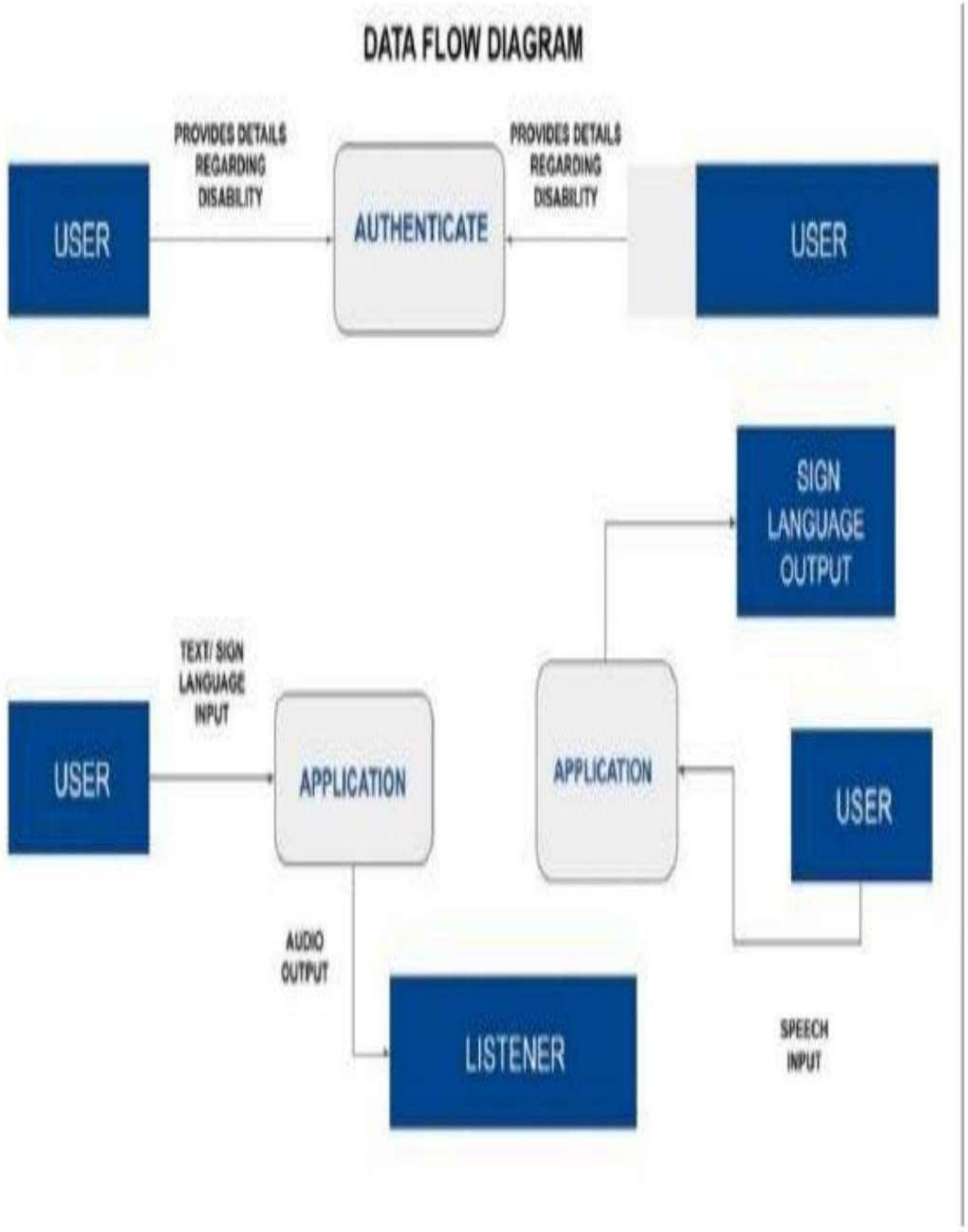
Non-functional Requirements:

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

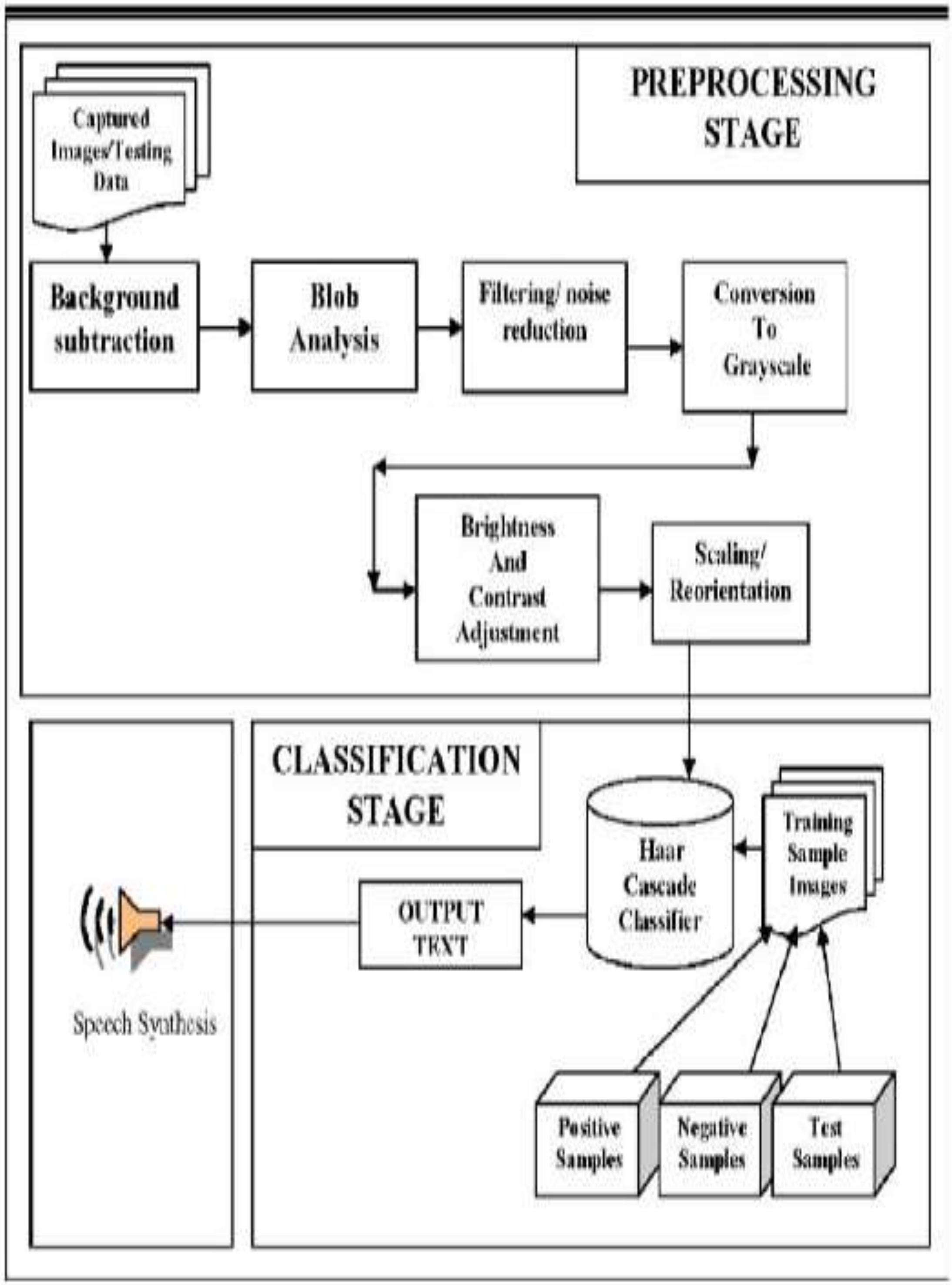
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Easy usable application for everyone.Especially useful for disabled person. It is user friendly
NFR-2	**Security**	It is also a secured application and information and images are securely stored.It must be ensured that the privacy of user 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-5	**Availability**	It is afree accessible and Universal access.Since sign language is a almost same everywhere ,the system can be used across the globe

5. PROJECT DESIGN

Data Flow Diagrams



Data flow diagram

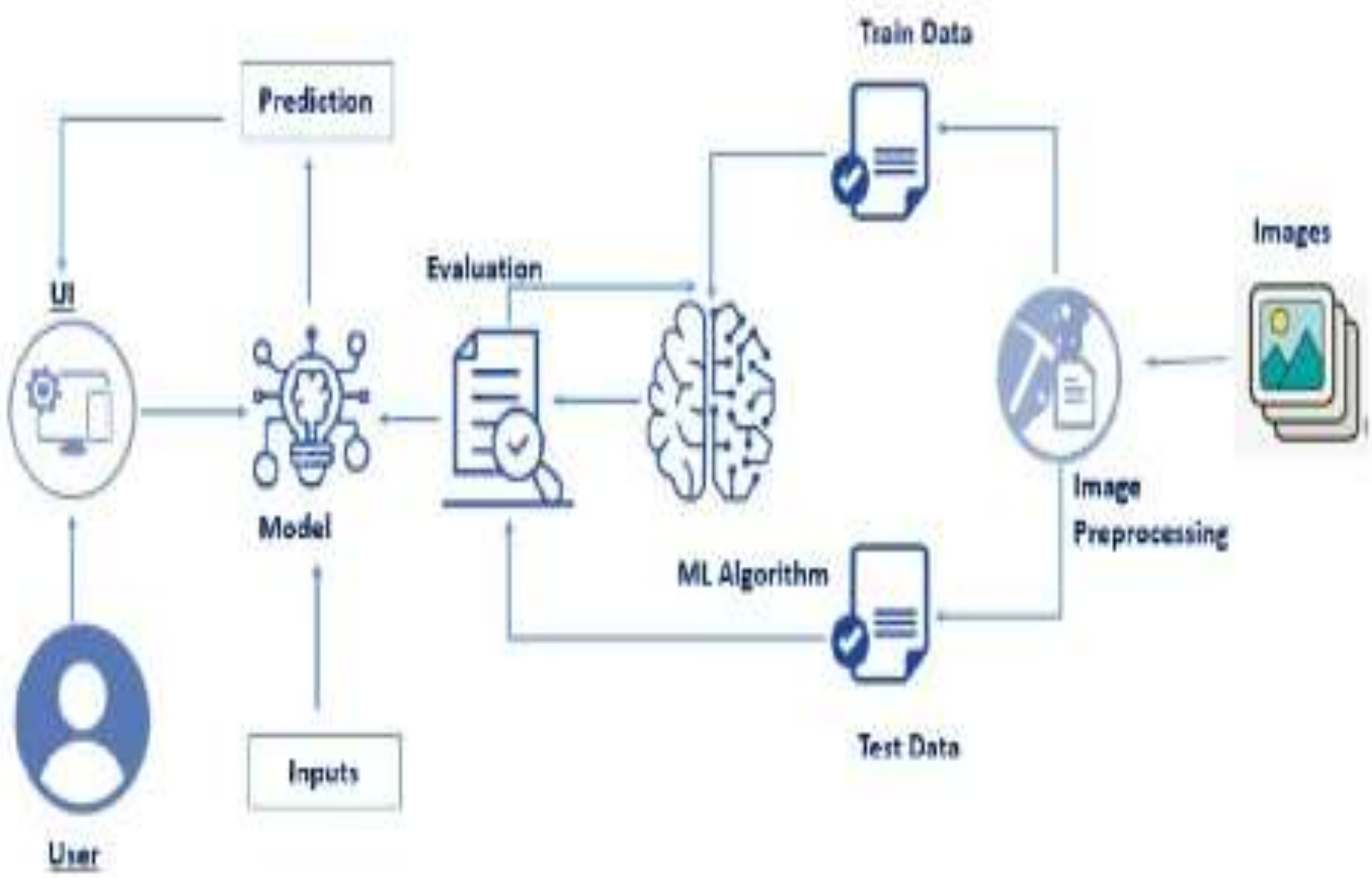


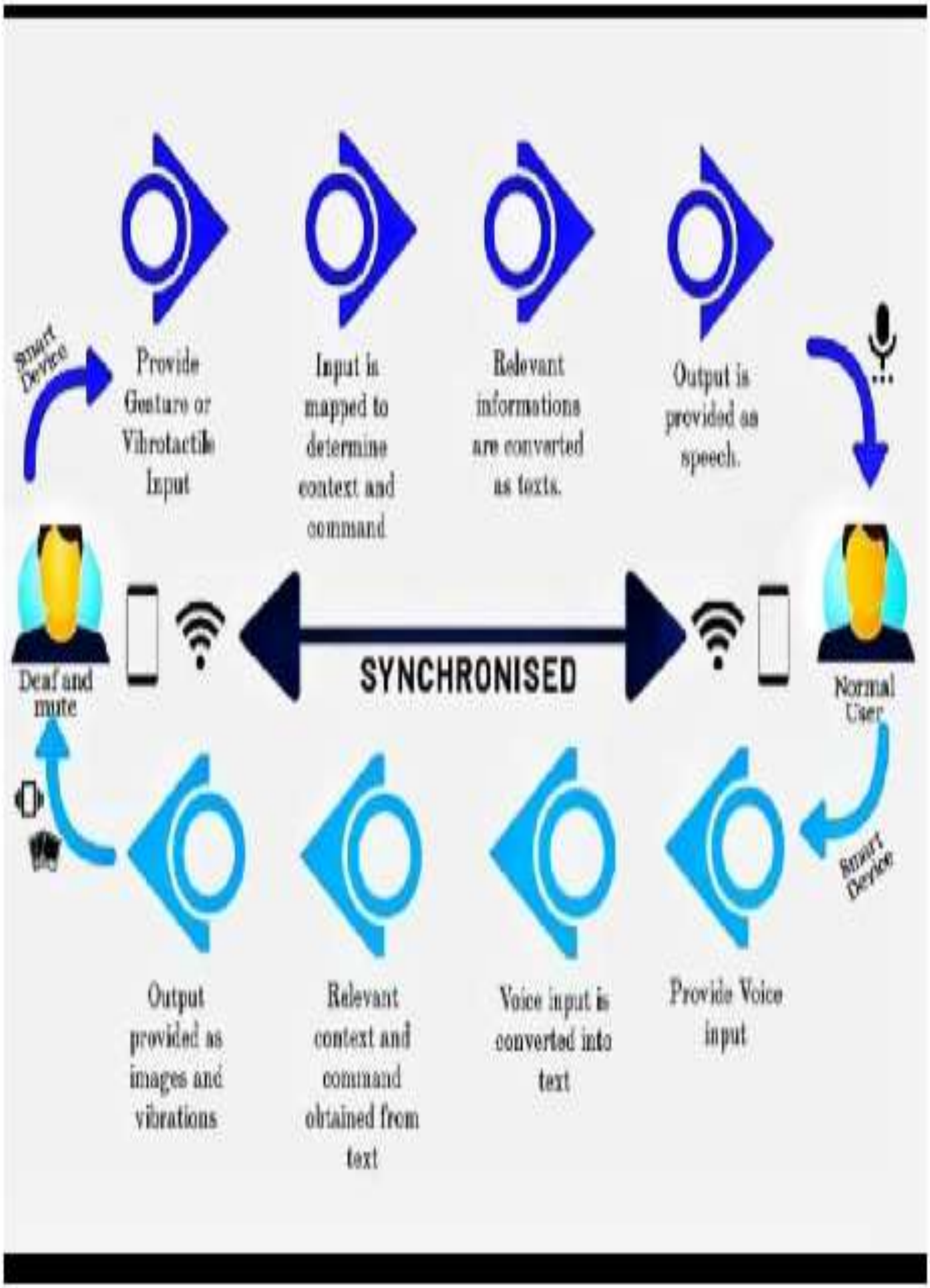
Solution Architecture:

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.

Example - Solution Architecture Diagram:




Smart Device
Provide Gesture or Vibrotactile Input
Input is mapped to determine context and command
Relevant informations are converted as texts.
Output is provided as speech.
Deaf and mute
SYNCHRONISED
Normal User
Output provided as images and vibrations
Relevant context and command obtained from text
Voice input is converted into text
Provide Voice input
Smart Device

Milestone Activity Plan.

Milestone	Function (Epic)	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 totrain 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 the 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 HTMLpages.
Milestone-6	Train CNN model	M6	Register for IBM Cloud and train ImageClassification Model.
Milestone-6	Final result	M7	To ensure all the activities and resulting thefinal output.

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	R.Aishwarya
Sprint-1		USN-2	Image preprocessing	8	Medium	S.Jayanthini V.Kanimozhi
Sprint-2	Model Building	USN-3	Import the required libraries, add the necessary layers and compile the 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 model's 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, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	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

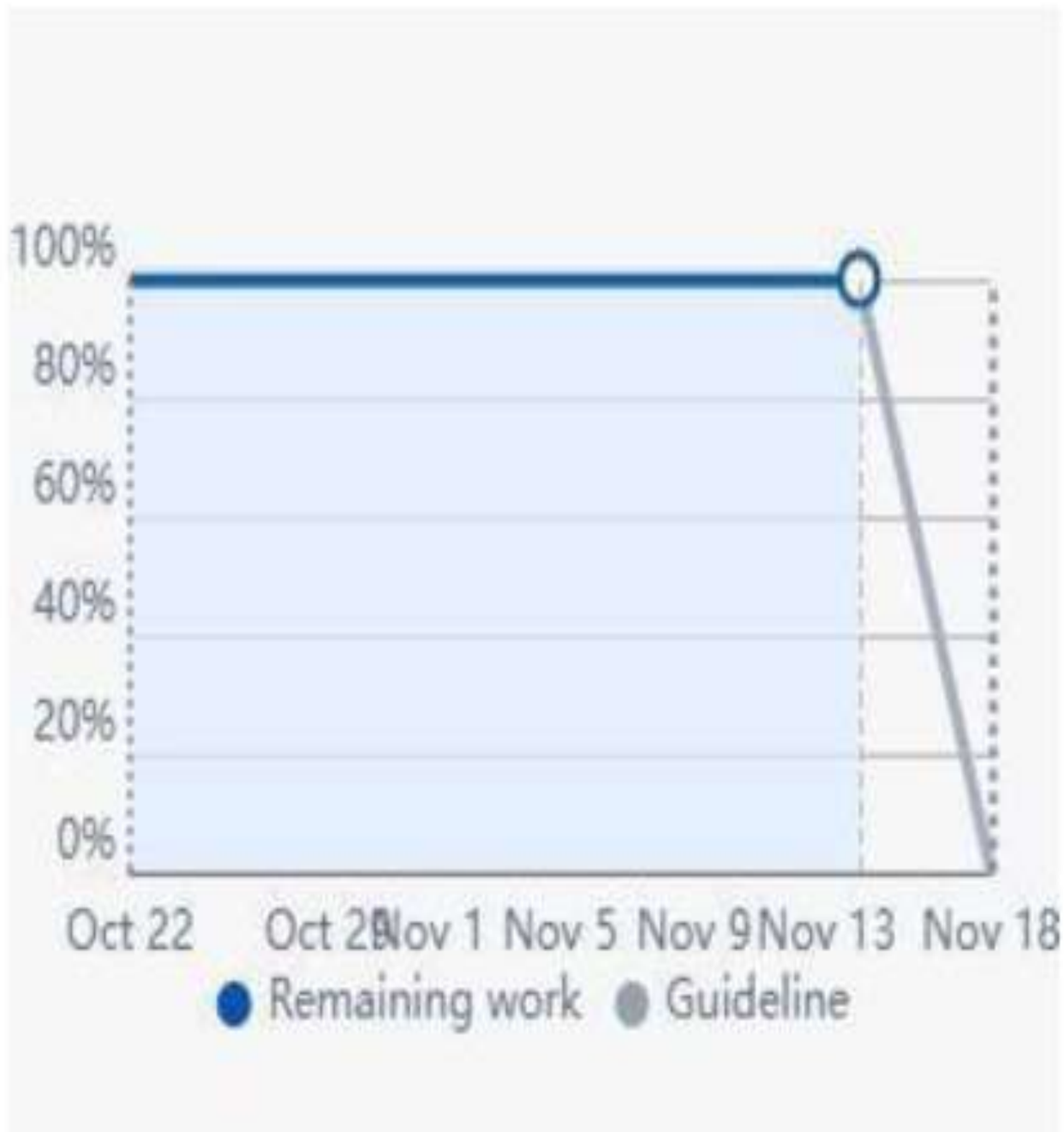
Velocity:

$$AV = \frac{sprint\ duration}{velocity}$$

AV = 6/10 = 0.6

Burndown chart:

SPRINT BURNDOWN CHART:



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

Model Building

Adding The Dense Layers

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

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

In [ ]:
```
print("Adding dense layer on top")
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

In [ ]:
```
print("Complete architecture of the model")
model.summary()
```

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

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

```
Len x-train :  18
```

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


```
Len x-train :  18
Len x-test :  3
```

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

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

Model Creation

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

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

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

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

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

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

Model Building

Adding The Flatten Layer

In [ ]:
```
# importing numpy as np
import numpy as np
```

In [ ]:
```
# declare flatten np
gfg = np.array([[6, 9, 12], [8, 5, 2], [18, 21, 24]])

# using array.flatten() method
flat_gfg = gfg.flatten(order='A')
print(flat_gfg)
```

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

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

```
Len x-train :  18
```

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

Model Creation

In [ ]:
```
model = Sequential()
for i, feat in enumerate(args.conv_f):
    if i==0:
        model.add(Conv2D(feat, input_shape=x[0].shape, kernel_size=3, padding = 'same',use_bias=False))
    else:
        model.add(Conv2D(feat, kernel_size=3, padding = 'same',use_bias=False))
        model.add(BatchNormalization())
        model.add(LeakyReLU(alpha=args.conv_act))
        model.add(Conv2D(feat, kernel_size=3, padding = 'same',use_bias=False))
        model.add(BatchNormalization())
        model.add(LeakyReLU(alpha=args.conv_act))
        model.add(Dropout(args.conv_do[i]))
```

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

#Input code here

denseArgs = {'use_bias':False}
for i,feat in enumerate(args.dense_f):
    model.add(Dense(feat,**denseArgs))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=args.dense_act))
    model.add(Dropout(args.dense_do[i]))
model.add(Dense(1))
```

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

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

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

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

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

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

Model Building

Adding The Pooling Layer

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

In [ ]:
```
import numpy as np
from keras.models import Sequential
from keras.layers import MaxPooling2D
```

In [ ]:
```
# define input image
image = np.array([[2, 2, 7, 3],
                  [9, 4, 6, 1],
                  [8, 5, 2, 4],
                  [3, 1, 2, 6]])
image = image.reshape(1, 4, 4, 1)
```

In [ ]:
```
# define model containing just a single max pooling layer
model = Sequential(
        [MaxPooling2D(pool_size = 2, strides = 2)])

# generate pooled output
output = model.predict(image)
```

In [ ]:
```
# print output image
output = np.squeeze(output)
print(output)
```

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

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

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

```
Len x-train :  18
Len x-test :  3
```

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

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

Model Creation

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

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

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

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

Model Building

Compile To The Model

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

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

In [ ]:
```
# Creating sample sourcecode to multiply two variables
# x and y.
srcCode = 'x = 10\ny = 20\nmul = x * y\nprint("mul =", mul)'

# Converting above source code to an executable
execCode = compile(srcCode, 'mulstring', 'exec')

# Running the executable code.
exec(execCode)
```

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
def compile_model_results(model, root="./"):

    listing = glob.glob(root + '/models/' + model + '/*/best_pars.pkl')

    dic_list = []
    for file in listing:
        tmp = hyper_parameters_load(file)
        dic_list.append(tmp.to_dictionary())

    df = pd.DataFrame(dic_list)
    df = pd.DataFrame(dic_list)
    df['diff'] = df.test_F1 - df.forecast_F1
    df['pci'] = abs(df.test_F1 - df.forecast_F1)

    if not os.path.exists(root + '/figures/' +  model ):
        os.makedirs(root + '/figures/' +  model )

    df.to_csv(root + '/figures/' +  model + '/results.csv', index=False)

    return df
```

In [ ]:
```
# Set optimizer loss and metrics
    opt = Adam(lr=args.initial_lr, beta_1=0.99, beta_2=0.999, decay=1e-6)
    if args.net.find('caps') != -1:
        metrics = {'out_seg': dice_hard}
    else:
        metrics = [dice_hard]

    loss, loss_weighting = get_loss(root=args.data_root_dir, split=args.split_num, net=args.net,
                                    recon_wei=args.recon_wei, choice=args.loss)

    # If using CPU or single GPU
    if args.gpus <= 1:
        uncomp_model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)
        return uncomp_model
    # If using multiple GPUs
    else:
        with tf.device("/cpu:0"):
            uncomp_model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)
            model = multi_gpu_model(uncomp_model, gpus=args.gpus)
            model.__setattr__('callback_model', uncomp_model)
        model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)

X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size,
random_state=seed)
```

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

```
Len x-train :  18
Len x-test :  3
```

Model Compilation

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

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

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

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

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

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

In [ ]:
```
# reading code from a file
f = open('main.py', 'r')
temp = f.read()
f.close()

code = compile(temp, 'main.py', 'exec')
exec(code)
```

Saving the Model

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

Model Building

Fit And Save The Model

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

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

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

```
model = LogisticRegression()
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))

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

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

```
Len x-train :  18
Len x-test :  3
```

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

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

Model Creation

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

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

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

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

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

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

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

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

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versi
on. Please use `Model.fit`, which supports generators.

Epoch 1/10
18/18 [==============================] - 92s 5s/step - loss: 0.0049 - accuracy: 0.9994 - val_loss: 0.2635 - val_accuracy: 0.9773
Epoch 2/10
18/18 [==============================] - 90s 5s/step - loss: 0.0040 - accuracy: 0.9995 - val_loss: 0.2074 - val_accuracy: 0.9773
Epoch 3/10
18/18 [==============================] - 87s 5s/step - loss: 0.0041 - accuracy: 0.9995 - val_loss: 0.2460 - val_accuracy: 0.9773
Epoch 4/10
18/18 [==============================] - 91s 5s/step - loss: 0.0041 - accuracy: 0.9992 - val_loss: 0.2470 - val_accuracy: 0.9782
Epoch 5/10
18/18 [==============================] - 88s 5s/step - loss: 0.0037 - accuracy: 0.9993 - val_loss: 0.2439 - val_accuracy: 0.9782
Epoch 6/10
18/18 [==============================] - 88s 5s/step - loss: 0.0024 - accuracy: 0.9997 - val_loss: 0.2852 - val_accuracy: 0.9782
Epoch 7/10
18/18 [==============================] - 91s 5s/step - loss: 0.0023 - accuracy: 0.9997 - val_loss: 0.2589 - val_accuracy: 0.9782
Epoch 8/10
18/18 [==============================] - 93s 5s/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.2523 - val_accuracy: 0.9782
Epoch 9/10
18/18 [==============================] - 92s 5s/step - loss: 0.0013 - accuracy: 0.9999 - val_loss: 0.2269 - val_accuracy: 0.9778
Epoch 10/10
18/18 [==============================] - 91s 5s/step - loss: 0.0012 - accuracy: 0.9999 - val_loss: 0.2968 - val_accuracy: 0.9782
```

Out[ ]:

Saving the Model

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

Model Building

Importing The Required Model Building Libraries

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

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

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

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

```
Len x-train :  18
Len x-test :  3
```

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

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

Model Creation

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

In [ ]:
```
dataset = pd.read_csv('E:\Datasets\Mall_Customers.csv')
```

Initializing The Model

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

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

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

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

```
Len x-train :  18
Len x-test :  3
```

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

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

Model Creation

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

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

8. TESTING

Test cases

Real-Time Communication System Powered By AI For Specially Abled

Loading the Dataset & Image Data Generation

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

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

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [4]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [5]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

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

Model Creation

In [6]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [7]:
```
# Creating Model
model=Sequential()
```

In [8]:
```
# Adding Layers
```

In [9]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [10]:
```
model.add(Flatten())
```

In [11]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [12]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [14]:
```
# 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))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future ver
on. Please use `Model.fit`, which supports generators.

Epoch 1/10
18/18 [==============================] - 97s 5s/step - loss: 0.0054 - accuracy: 0.9991 - val_loss: 0.3700 - val_accuracy: 0.9756
Epoch 2/10
18/18 [==============================] - 97s 5s/step - loss: 0.0039 - accuracy: 0.9996 - val_loss: 0.3347 - val_accuracy: 0.9751
Epoch 3/10
18/18 [==============================] - 95s 5s/step - loss: 0.0036 - accuracy: 0.9996 - val_loss: 0.3324 - val_accuracy: 0.9756
Epoch 4/10
18/18 [==============================] - 94s 5s/step - loss: 0.0033 - accuracy: 0.9996 - val_loss: 0.3712 - val_accuracy: 0.9747
Epoch 5/10
18/18 [==============================] - 95s 5s/step - loss: 0.0033 - accuracy: 0.9995 - val_loss: 0.3011 - val_accuracy: 0.9764
Epoch 6/10
18/18 [==============================] - 95s 5s/step - loss: 0.0028 - accuracy: 0.9997 - val_loss: 0.2759 - val_accuracy: 0.9769
Epoch 7/10
18/18 [==============================] - 94s 5s/step - loss: 0.0024 - accuracy: 0.9997 - val_loss: 0.3056 - val_accuracy: 0.9769
Epoch 8/10
18/18 [==============================] - 95s 5s/step - loss: 0.0021 - accuracy: 0.9997 - val_loss: 0.3332 - val_accuracy: 0.9760
Epoch 9/10
18/18 [==============================] - 93s 5s/step - loss: 0.0019 - accuracy: 0.9997 - val_loss: 0.3236 - val_accuracy: 0.9760
Epoch 10/10
18/18 [==============================] - 93s 5s/step - loss: 0.0016 - accuracy: 0.9997 - val_loss: 0.3429 - val_accuracy: 0.9760
```

Out[14]:

Saving the Model

In [15]:
```
model.save('asl_model_84_54.h5')
```

Testing the model

In [16]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [18]:
```
model=load_model('asl_model_84_54.h5')
img=image.load_img(r'/content/drive/MyDrive/Dataset/test_set/D/2.png',
                   target_size=(64,64))
```

Real-Time Communication System Powered By AI For Specially Abled

Loading the Dataset & Image Data Generation

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

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

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

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [4]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [5]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

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

Model Creation

In [6]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [7]:
```
# Creating Model
model=Sequential()
```

In [8]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [8]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [9]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [10]:
```
model.add(Flatten())
```

In [11]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [12]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [14]:
```
# 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))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versi
on. Please use `Model.fit`, which supports generators.

Epoch 1/10
18/18 [==============================] - 97s 5s/step - loss: 0.0054 - accuracy: 0.9991 - val_loss: 0.3700 - val_accuracy: 0.9756
Epoch 2/10
18/18 [==============================] - 97s 5s/step - loss: 0.0039 - accuracy: 0.9996 - val_loss: 0.3347 - val_accuracy: 0.9751
Epoch 3/10
18/18 [==============================] - 95s 5s/step - loss: 0.0036 - accuracy: 0.9996 - val_loss: 0.3324 - val_accuracy: 0.9756
Epoch 4/10
18/18 [==============================] - 94s 5s/step - loss: 0.0033 - accuracy: 0.9996 - val_loss: 0.3712 - val_accuracy: 0.9747
Epoch 5/10
18/18 [==============================] - 95s 5s/step - loss: 0.0033 - accuracy: 0.9995 - val_loss: 0.3011 - val_accuracy: 0.9764
Epoch 6/10
18/18 [==============================] - 95s 5s/step - loss: 0.0028 - accuracy: 0.9997 - val_loss: 0.2759 - val_accuracy: 0.9769
Epoch 7/10
18/18 [==============================] - 94s 5s/step - loss: 0.0024 - accuracy: 0.9997 - val_loss: 0.3056 - val_accuracy: 0.9769
Epoch 8/10
18/18 [==============================] - 95s 5s/step - loss: 0.0021 - accuracy: 0.9997 - val_loss: 0.3332 - val_accuracy: 0.9760
Epoch 9/10
18/18 [==============================] - 93s 5s/step - loss: 0.0019 - accuracy: 0.9997 - val_loss: 0.3236 - val_accuracy: 0.9760
Epoch 10/10
18/18 [==============================] - 93s 5s/step - loss: 0.0016 - accuracy: 0.9997 - val_loss: 0.3429 - val_accuracy: 0.9760
```

Out[14]:

Saving the Model

In [15]:
```
model.save('asl_model_84_54.h5')
```

Testing the model

In [16]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

In [18]:
```
model=load_model('asl_model_84_54.h5')
img=image.load_img(r'/content/drive/MyDrive/Dataset/test_set/D/2.png',
                   target_size=(64,64))
```

In [19]:
```
img
```

Out[19]:

In [20]:
```
x=image.img_to_array(img)
```

In [21]:
```
x.ndim
```

Out[21]: 3

In [22]:
```
x=np.expand_dims(x,axis=0)
```

In [23]:
```
x.ndim
```

Out[23]: 4

In [24]:
```
pred=np.argmax(model.predict(x),axis=1)
```

```
1/1 [==============================] - 0s 145ms/step
```

In [25]:
```
pred
```

In [25]:
```
pred
```

Out[25]: array([3])

In [26]:
```
index=['A','B','C','D','E','F','G','H','I']
print(index[pred[0]])
```

```
D
```

OPEN CV

In [27]:
```
import cv2
```

In [30]:
```
img=cv2.imread(r'/content/drive/MyDrive/Dataset/test_set/C/2.png',1)
```

In [31]:
```
img1=cv2.imread(r'/content/drive/MyDrive/Dataset/test_set/B/2.png',0)
```

In [32]:
```
print(img.shape)
```

```
(64, 64, 3)
```

In [40]:
```
from google.colab.patches import cv2_imshow
cv2_imshow(img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

User Acceptance Testing

Date	14 November 2022
Team ID	PNT2022TMID28925
Project Name	Project – Real time systems powered by AI for specially abled
Maximum Marks	4 Marks

1. PurposeofDocument

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

2. DefectAnalysis

Thisreportshowsthenumberofresolvedor closed bugs at each severity level, and how they were resolved

Resolution	Severity1	Severity2	Severity3	Severity4	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'tFix	0	4	1	1	7
Totals	27	17	14	27	86

3. TestCaseAnalysis

Thisreportshowsthenumberoftestcasesthathavepassed,failed,anduntested

Section	TotalCases	Not Tested	Fail	Pass
PrintEngine	8	0	0	8
ClientApplication	49	0	0	49
Security	4	0	0	4
OutsourceShipping	4	0	0	4
ExceptionReporting	11	0	0	11
FinalReportOutput	2	0	0	2
VersionControl	1	0	0	1

9. RESULTS

Performance Metrices

	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation?
4	LoginPage_TC_003	Functional	Home Page	Verify user is able to use the Login/Signup popup when user clicked on My account button		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify Login/Signup popup displayed or not	https://rkfi.powered-by-firstspirit.com/	Login/Signup popup should display	Working as expected	Pass		
5	LoginPage_TC_003	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify Login/Signup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Don't remember? Recovery password link	https://rkfi.powered-by-firstspirit.com/	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Don't remember? Recovery password link	Working as expected	pass		
6	LoginPage_TC_003	Functional	Home page	Verify user is able to login into application with Valid credentials		1.Enter URL(https://rkfi.powered-by-firstspirit.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on Login button	Username: qaRK7@ password: 3980003	User should navigate to user account homepage	Working as expected	pass		
7	LoginPage_TC_004	Functional	Login page	Verify user is able to login into application with Valid credentials		1.Enter URL(https://rkfi.powered-by-firstspirit.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on Login button	Username: qaRK7@ password: 3980003	Application should show 'Incorrect email or password!' validation message	Working as expected	Pass		
8	LoginPage_TC_004	Functional	Login page	Verify user is able to login into application with Valid credentials		1.Enter URL(https://rkfi.powered-by-firstspirit.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter invalid password in password text box 5.Click on Login button	Username: qaRK7@ password: 3980003	Application should show 'Incorrect email or password!' validation message	Working as expected	pass		
9	LoginPage_TC_005	Functional	Login page	Verify user is able to login into application with Valid credentials		1.Enter URL(https://rkfi.powered-by-firstspirit.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter invalid password in password text box 5.Click on Login button	Username: qaRK7@ password: 3980003	Application should show 'Incorrect email or password!' validation message	Working as expected	pass		
10	LoginPage_TC_006	Functional	Login page	Verify user is able to use login page		1.Verify that cursor is focused on the "Username" text box on the page load (login page) 2.Verify that tab flow functility is working properly or not. 3.Verify that Enter/Tab key works as a substitute for the Signin button. 4.Verify that the User is able to login with Valid Credentials.	Username: qaRK7@ password: 3980003	User can use the login page	Working as expected	pass		
11	LoginPage_TC_007	Functional	Login page	Verify user is able to login into application or not?		1.Verify that cursor is focused on the "Username" text box on the page load (login page) 2.Verify that tab flow functility is working properly or not. 3.Verify that Enter/Tab key works as a substitute for the Signin button. 4.verify that user is able to login into application	Username: qaRK7@ pass word: 3980003	User could be login into the application	Working as expected	Pass		
12	LoginPage_TC_008	Functional	Login page	Verify user is able to navigate to create your account page?		1.Verify that cursor is focused on the "Username" text box on the page load (login page) 2.Verify that Enter/Tab key works as a substitute for the Signin button. 3.verify that user is able to login into application 4.verify that user is able to navigate to the next page	Username: qaRK7@ pass word: 3980003	User can navigate to the account page to create an account	Working as expected	pass		
13	LoginPage_TC_009	Functional	Login page	Verify user is able to recovery password		1.Verify that cursor is focused on the "Username" text box on the page load (login page) 2.Verify that Enter/Tab key works as a substitute for the Signin button. 3.verify that user is able to enter email id and password 4.verify that user is able to recover password by using forgot password	Username: qaRK7@ pass word: 3980003	User can able to recover the password	Working as expected	pass		
14	LoginPage_TC_010	Functional	Login page	Verify login page elements		1.Verify that cursor is focused on the "Username" text box on the page load (login page) 2.Verify that Enter/Tab key works as a substitute for the Sign in button 3. verify that user is able to sign up 4.verify that user is able to enter the login elements like password email id etc	Username: qaRK7@ pass word: 3980003	User can able to verify the login elements in the page		pass		
15	Search_TC_01	Functional	Search page	Verify user is able to search by entering keyword in search box		1.Verify that cursor is focused on the "Username" text box on the page load (login page) 2.Verify that Enter/Tab key works as a substitute for the Sign in button 3. verify that user is able to search box 4.verify user is able to enter keyword in search box	Username: qaRK7@ pass word: 3980003	User is able to search an word by entering keyword	Working as expected	pass		
16	Search_TC_02	Functional	Search page	Verify user is able to see suggestions based on keyword entered in search box		1.Verify that cursor is focused on the "Username" text box on the page load (login page) 2.Verify that Enter/Tab key works as a substitute for the Sign in button 3. verify that user is able to search in search box 4.verify that user is able to see keyword while typing in search box based on their suggestion	Username: qaRK7@ pass word: 3980003	User can able to see suggestion when entering in search box	Working as expected	pass		
17	Search_TC_03	Functional	Search page	Verify user is able to see relevant auto suggestions displaying based on keyword entered in search box		1.Verify that cursor is focused on the "Username" text box on the page load (login page) 2.Verify that Enter/Tab key works as a substitute for the Sign in button 3. verify that user is able to search in search box 4.verify that user is able to see auto suggestions while typing	Username: qaRK7@ pass word: 3980003	User can able to see auto suggestions while entering the input events	Working as expected	pass		

10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES

It enables employees from across the world to communicate with each other 24×7 and share ideas or solve problems quickly. It is a cost-effective way of getting several people from different locations to attend meetings and conferences – without having to spend time or money on travel, and accommodation.

DISADVANTAGES

The biggest disadvantage of communication is that it takes a lot of time to listen, speak, read, or write to someone. While trying to do one thing you can accidentally hurt another person's feelings by not listening or paying attention. This could result in damaging your relationship with them.

11. CONCLUSION

Real-time communication (RTC) workloads can be deployed on AWS to attain scalability, elasticity, and high availability while meeting the key requirements. Today, several customers are using AWS, its partners, and open source solutions to run RTC workloads with reduced cost and faster agility as well as a reduced global footprint.The reference architectures and best practices provided in this white paper can help customers successfully set up RTC workloads on AWS and optimize the solutions to meet end user requirements while optimizing for the cloud.

12. FUTURE SCOPE

1. Through image recognition technology, AI understands the context of objects in photos and describes photos to people.

2. The speech-to-text and text-to-speech technologies helped those people who had speech impediments

3. The product in AI that narrates the entire world around them visually impaired by reading texts, describing whereabouts and the looks of the nearby people by identifying and recognizing faces and emotions.

4. Autonomous vehicles are in trend and their success is due to AI technology. These vehicles can be beneficial to people living with limited physical mobility

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()
```

```
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()
```

```html
<!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: #04AA6D;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
}
```

```
28
29  button:hover {
30    opacity: 0.8;
31  }
32
33  /* Extra styles for the cancel button */
34  .cancelbtn {
35    width: auto;
36    padding: 10px 18px;
37    background-color: #f44336;
38  }
39
40  /* Center the image and position the close button */
41  .imgcontainer {
42    text-align: center;
43    margin: 24px 0 12px 0;
44    position: relative;
45  }
46
47  img.avatar {
48    width: 40%;
49    border-radius: 50%;
50  }
51
52  .container {
53    padding: 16px;
```

```
55
56  span.psw {
57    float: right;
58    padding-top: 16px;
59  }
60
61  /* The Modal (background) */
62  .modal {
63    display: none; /* Hidden by default */
64    position: fixed; /* Stay in place */
65    z-index: 1; /* Sit on top */
66    left: 0;
67    top: 0;
68    width: 100%; /* Full width */
69    height: 100%; /* Full height */
70    overflow: auto; /* Enable scroll if needed */
71    background-color: rgb(0,0,0); /* Fallback color */
72    background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
73    padding-top: 60px;
74  }
75
76  /* Modal Content/Box */
77  .modal-content {
78    background-color: #fefefe;
79    margin: 5% auto 15% auto; /* 5% from the top, 15% from the bottom and centered */
80    border: 1px solid #888;
81    width: 80%; /* Could be more or less, depending on screen size */
82  }
```

```
82  }
83
84  /* The Close Button (x) */
85  .close {
86    position: absolute;
87    right: 25px;
88    top: 0;
89    color: #000;
90    font-size: 35px;
91    font-weight: bold;
92  }
93
94  .close:hover,
95  .close:focus {
96    color: red;
97    cursor: pointer;
98  }
99
100 /* Add Zoom Animation */
101 .animate {
102   -webkit-animation: animatezoom 0.6s;
103   animation: animatezoom 0.6s
104 }
105
106 @-webkit-keyframes animatezoom {
107   from {-webkit-transform: scale(0)}
108   to {-webkit-transform: scale(1)}
109 }
110
111 @keyframes animatezoom {
112   from {transform: scale(0)}
113   to {transform: scale(1)}
114 }
115
116 /* Change styles for span and cancel button on extra small screens */
117 @media screen and (max-width: 300px) {
118   span.psw {
119      display: block;
120      float: none;
121   }
122   .cancelbtn {
123      width: 100%;
124   }
125 }
126 </style>
127 </head>
128 <body>
129
130 <h2>REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED</h2>
131
132 <button onclick="document.getElementById('id01').style.display='block'" style="width:auto;">Login</button>
133
134 <div id="id01" class="modal">
135
136   <form class="modal-content animate" action="/action_page.php" method="post">
137     <div class="imgcontainer">
```

```
138         <span onclick="document.getElementById('id01').style.display='none'" class="close" title="Close Modal">&times;</span>
139         <img src="https://static.vecteezy.com/system/resources/thumbnails/007/407/996/small/user-icon-person-icon-client-symbol-login-head-sign-icon-design-vec
140       </div>
141
142       <div class="container">
143         <label for="uname"><b>Username</b></label>
144         <input type="text" placeholder="Enter Username" name="uname" required>
145
146         <label for="psw"><b>Password</b></label>
147         <input type="password" placeholder="Enter Password" name="psw" required>
148
149         <button type="submit">Login</button>
150         <label>
151           <input type="checkbox" checked="checked" name="remember"> Remember me
152         </label>
153       </div>
154
155       <div class="container" style="background-color:#f1f1f1">
156         <button type="button" onclick="document.getElementById('id01').style.display='none'" class="cancelbtn">Cancel</button>
157         <span class="psw">Forgot <a href="#">password?</a></span>
158       </div>
159     </form>
160   </div>
161   <!doctype html>
162   <html lang="en">
163   <head>
164       <meta charset="UTF-8">
165       <meta name="viewport"
```



```
166             content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
167       <meta http-equiv="X-UA-Compatible" content="ie=edge">
168       <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
169       <link rel="stylesheet" href="style.css">
170       <title>Document</title>
171   </head>
172   <body>
173   <div class="display-cover">
174       <video autoplay></video>
175       <canvas class="d-none"></canvas>
176
177       <div class="video-options">
178           <select name="" id="" class="custom-select">
179               <option value="">Select camera</option>
180           </select>
181       </div>
182
183       <img class="screenshot-image d-none" alt="">
184
185       <div class="controls">
186           <button class="btn btn-danger play" title="Play"><i data-feather="play-circle"></i></button>
187           <button class="btn btn-info pause d-none" title="Pause"><i data-feather="pause"></i></button>
188           <button class="btn btn-outline-success screenshot d-none" title="ScreenShot"><i data-feather="image"></i></button>
189       </div>
190   </div>
191
192   <script src="https://unpkg.com/feather-icons"></script>
193   <script src="script.js"></script>
```

```
194 </body>
195 <html><head>
196 </head><body>
197     <video src="" ></video>
198     <br />
199 <button id='flipCamera'>Flip</button>
200 </body>
201 <script>
202   var front = false;
203 var video = document.querySelector('video');
204   document.getElementById('flipCamera').onclick = function() { front = !front; };
205   var constraints = { video: { facingMode: (front? "user" : "environment"), width: 640, height: 480  } };
206   navigator.mediaDevices.getUserMedia(constraints)
207   .then(function(mediaStream) {
208     video.srcObject = mediaStream;
209     video.onloadedmetadata = function(e) {
210     video.play();
211 };
212 })
213 .catch(function(err) { console.log(err.name + ": " + err.message); })
214 </script></html>
215 </html>
216 <style>
217 .screenshot-image {
218     width: 150px;
219     height: 90px;
220     border-radius: 4px;
221     border: 2px solid whitesmoke;
222     box-shadow: 0 1px 2px 0 rgba(0, 0, 0, 0.1);
223     position: absolute;
224     bottom: 5px;
225     left: 10px;
226     background: white;
227 }
228
229 .display-cover {
230     display: flex;
231     justify-content: center;
232     align-items: center;
233     width: 70%;
234     margin: 5% auto;
235     position: relative;
236 }
237
238 video {
239     width: 100%;
240     background: rgba(0, 0, 0, 0.2);
241 }
242
243 .video-options {
244     position: absolute;
245     left: 20px;
246     top: 30px;
247 }
248
249 .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 #D2002E;
}

.controls button:nth-child(1) svg {
    color: #D2002E;
}

.controls button:nth-child(2) {
    border: 2px solid #008496;
}

.controls button:nth-child(2) svg {
    color: #008496;
}

.controls button:nth-child(3) {
    border: 2px solid #00B541;
}
```

```
307  .controls button:nth-child(3) svg {
308      color: #00B541;
309  }
310
311  .controls > button {
312      width: 45px;
313      height: 45px;
314      text-align: center;
315      border-radius: 100%;
316      margin: 0 6px;
317      background: transparent;
318  }
319
320  .controls > button:hover svg {
321      color: white;
322  }
323  </style>
324
325  <script>
326  // Get the modal
327  var modal = document.getElementById('id01');
328
329  // When the user clicks anywhere outside of the modal, close it
330  window.onclick = function(event) {
331      if (event.target == modal) {
332          modal.style.display = "none";
333      }
334  }
335  feather.replace();
```

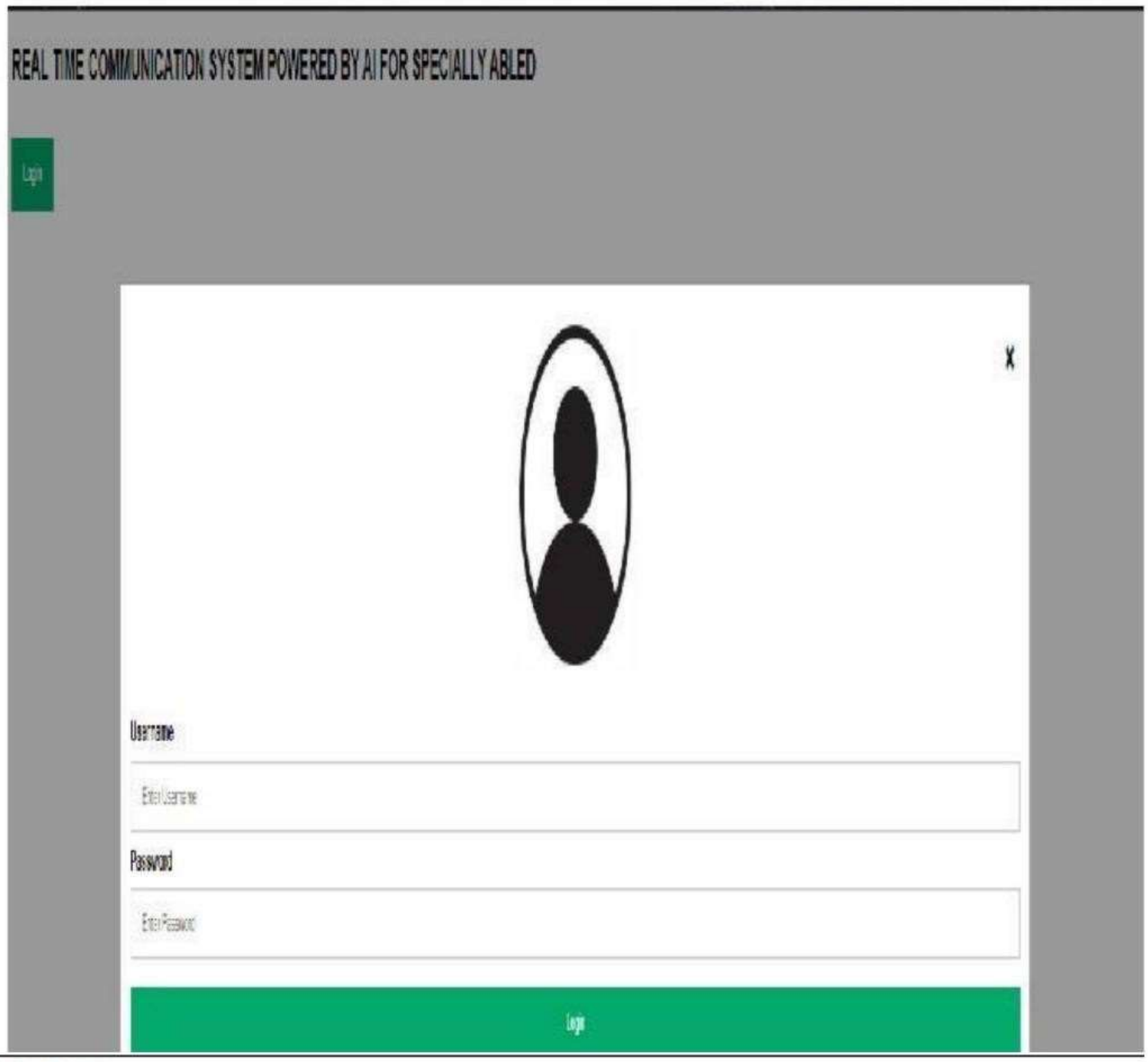
```

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);
  }
};
```

```
394     const stream = await navigator.mediaDevices.getUserMedia(constraints);
395     handleStream(stream);
396   };
397
398   const handleStream = (stream) => {
399     video.srcObject = stream;
400     play.classList.add('d-none');
401     pause.classList.remove('d-none');
402     screenshot.classList.remove('d-none');
403     streamStarted = true;
404   };
405
406   getCameraSelection();
407   ...
408   cameraOptions.onchange = () => {
409     const updatedConstraints = {
410       ...constraints,
411       deviceId: {
412         exact: cameraOptions.value
413       }
414     };
415     startStream(updatedConstraints);
416   };
417
418   const pauseStream = () => {
419     video.pause();
420     play.classList.remove('d-none');
421     pause.classList.add('d-none');
```



```
422   };
423
424   const doScreenshot = () => {
425     canvas.width = video.videoWidth;
426     canvas.height = video.videoHeight;
427     canvas.getContext('2d').drawImage(video, 0, 0);
428     screenshotImage.src = canvas.toDataURL('image/webp');
429     screenshotImage.classList.remove('d-none');
430   };
431
432   pause.onclick = pauseStream;
433   screenshot.onclick = doScreenshot;
434   </script>
435
436   </body>
437   </html>
```




```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 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>
```

```html
        </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
.fit-cover {
  object-fit: cover;
}
```

```
.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%;
}
```