IBM - Nalaiya Thiran

Real-Time Communication System Powered by AI for Specially Abled

TEAM ID : PNT2022TMID42702

GROUP MEMBERS :

NAME	REG NO:
SASIDHARAN N	711219104025
SUJAN S	711219104030
VISHVA B	711219104033
SIVANADHAN	711219104027
SIVASURYA M	711219104704

1.INTRODUCTION

1.1 Project Overview

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 challenging 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 beneficial to have a proper conversation between a normal person and an impaired person in any language.

1.2 Purpose

The projects purpose is to create a system that convert 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.

It can generate revenue through direct customers and collaborate with health care sector and generate revenue from their customers. The main purpose of this application is to make deafmute people feel independent and more confident

2. LITERATURE SURVEY

2.1 Existing problem

Some of the real time existing solutions for solving this problem are:

Interpreter

If a sign language interpreter is available, this facilitates easy communication if the person who is deaf is fluent in sign language. The deaf person and person who is blind can communicate with each other via the interpreter. The deaf person can use sign language and the interpreter can speak what has been said to the person who is blind and then translate anything spoken by the blind person into sign language for the deaf person.

Technology

One of the easiest ways to communicate is through technology such as a smart phone or laptop. A deaf person can type out what they want to say and a person who is blind or has low vision can use a screen reader to read the text out loud. A blind person can also use voice recognition software to convert what they are saying in to text so that a person who is Deaf can then read it.

Just Speaking

Depending on the deaf person's level of hearing loss, they may be able to communicate with a blind person who is using speech. For example, a deaf person may have enough residual hearing (with or without the use of an assistive hearing device such as a hearing aid) to be able to decipher the speech of the person who is blind or has low vision. However, this is often not the most effective form of communication, as it is very dependent on the individual circumstances of both people and their environment (for example, some places may have too much background noise)

2.2 References

1) Title: Full Duplex Communication System for Deaf & Dumb People
   Author: Shraddha R. Ghorpade, Surendra K. Waghamare
   Year: 2015

2) Title: Sign Language Recognition System
   Author: Er. Aditi Kalsh, Dr N.S. Garewal
   Year: 2013

3) Title: Intelligent Sign Language Recognition Using Image Processing
   Author: Sawant Pramada, Deshpande Saylee , Nale Pranita, Nerkar Samiksha, Mrs. Archana, S. Vaidya
   Year: 2013

4) Title: AAWAAZ: A Communication System for Deaf & Dumb
   Author: Anchal Sood, Anju Mishra
   Year: 2016

2.3 Problem Statement Definition

A World Health Organization report says around 63 million people in India suffer from either complete or partial deafness, and of these, at least 50 lakh are children. 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 in emergency times as well as in normal times.

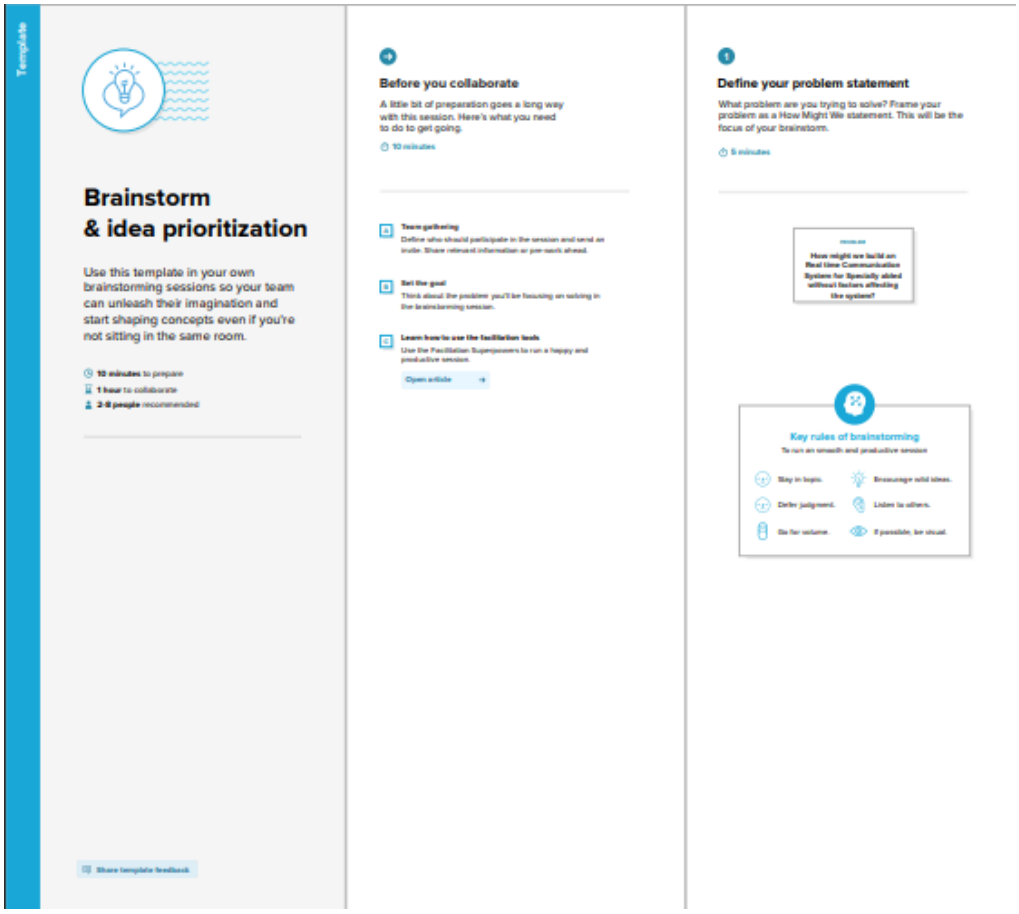
3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy map canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.
It is a useful tool to helps teams better understand their users.

3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.



2 Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Sasidharan N

- It makes people to communicate with Specially abled.
- In emergency times conveying their message is very dificult
- Our objective is to blend deaf and dumb within society and make them able to use their personal computers more effectively and efficiently
- User Friendly
- Neural networks can solve somewhat complicated issues at a mash easier level concerning the complexity of algorithms

SivaNadhan

- Ai Agorithms are used
- speak directly to the person rather than the person with them
- only refer to the person's disability if necessary or relevant
- avoid saying anything that implies the person with disability is superhuman, courageous or specia
- Higher Accuracy

Sujan S

- Quick Response
- Accurate understanding of how the person tries to communicate
- Gives the best Result
- It is very dificult for mute people to convey their message to normal people.
- Gives the best Result

Vishva B

- Communication should be universal without any barriers or limitations
- develop new forms of treatment, accelerate and improve patient care, and better manage electronic
- avoid saying anything that implies the person with disability is superhuman, courageous
- Cost - Free
- Trustworthy

3 Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Model Idea

Detailed UI which is user friendly. Flaks are stored and model is trained using AI algorithm. It is safe and secure.

Technology

AI uses it's potential to develop solutions to many physical and cognitive challenges For disabled people Face at work and daily life to promote social inclusion for them

Solution Approched

pre trained models are used to communicate with disabled people. And deep learning is used to solve the problem.

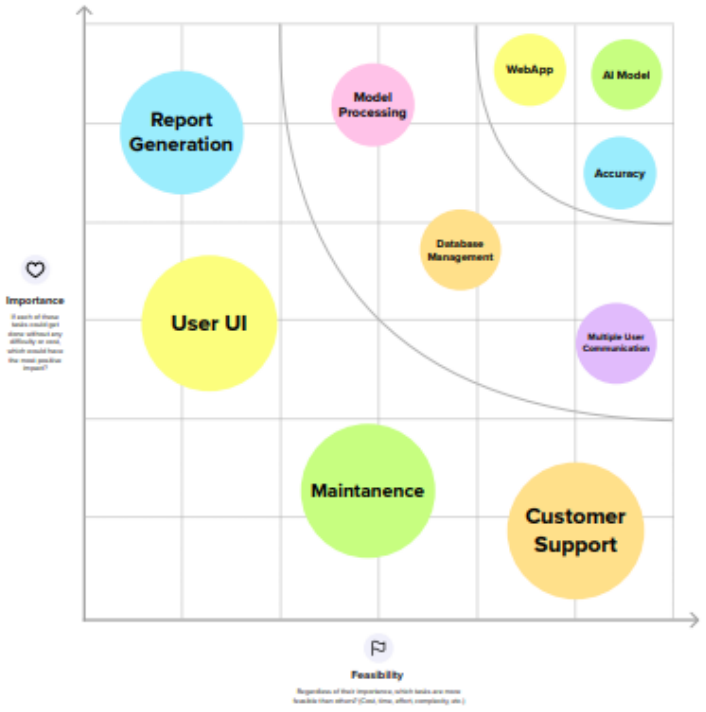
Innovation

Increased morale and corporate culture. Helps the disabled people and can show their skills and talents. Enhances an Organization's Reputation, Brand

4 Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

- **Share the mural**
  Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- **Export the mural**
  Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

- **Strategy blueprint**
  Define the components of a new idea or strategy.
  Open the template →
- **Customer experience journey map**
  Understand customer needs, motivations, and obstacles for an experience.
  Open the template →
- **Strengths, weaknesses, opportunities & threats**
  Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
  Open the template →

Share template feedback

3.3 Proposed solution

The project aims to develop a system that overcomes the problem faced by the speech and hearing impaired. The objectives of the research are as follow:

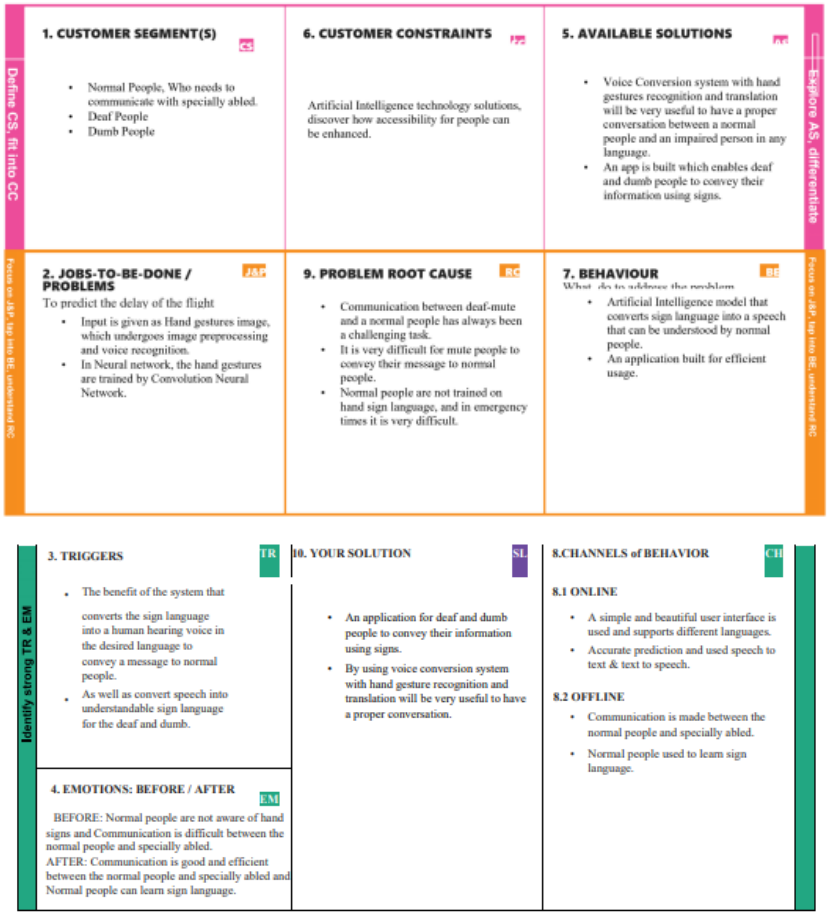
- ➤ To design and develop a system which lowers the communication gap between speech hearing impaired and normal world.

- ➤ To build a communication system that enables communications between deaf-dumb person and a normal person.

- ➤ A convolution neural network to create a model that is trained on different hand gestures. This model is used to create an app. 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.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To Model a System for aiding deaf dumb people and help them to communicate with a normal person in real world situation.
2.	Idea / Solution description	Using AI Technology this problem can be solved by importing various kinds of libraries in python code for the Model development.
3.	Novelty / Uniqueness	For Speaking and Listening(ASL) it has a rich Vocabulary and Grammar unique from any Spoken Languages.
4.	Social Impact / Customer Satisfaction	This allows to help to bridge the gaps in communication with hearing and speaking impaired people.
5.	Business Model (Revenue Model)	The app can be made available to more groups which will increase its growth.
6.	Scalability of the Solution	It will bring a new evolution in Real Time Communication System Powered by AI for Specially Abled with less time and safe enough resources.

3.4 Problem Solution Fit



4. REQUIREMENT ANALYSIS

4.1 Functional Requirement

Following are the functional requirements of the proposed solution.

FR. NO.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR - 1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR -2	User Confirmation	Confirmation via Email Confirmation via OTP
FR - 3	Uploading image	Upload image through camera or Upload image through gallery

FR - 4	Templates usage during any emergencies	Select emergency templates icon to pass the message quickly
FR - 5	Text to speech	Converts respective text to sign language

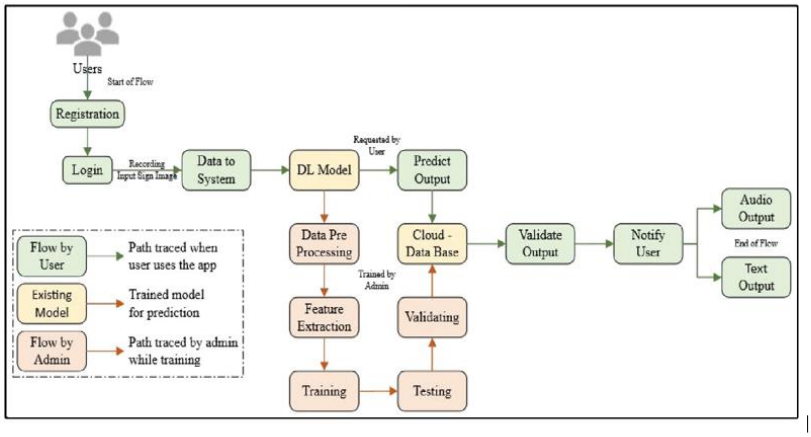
4.2 Non Functional Requirement

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

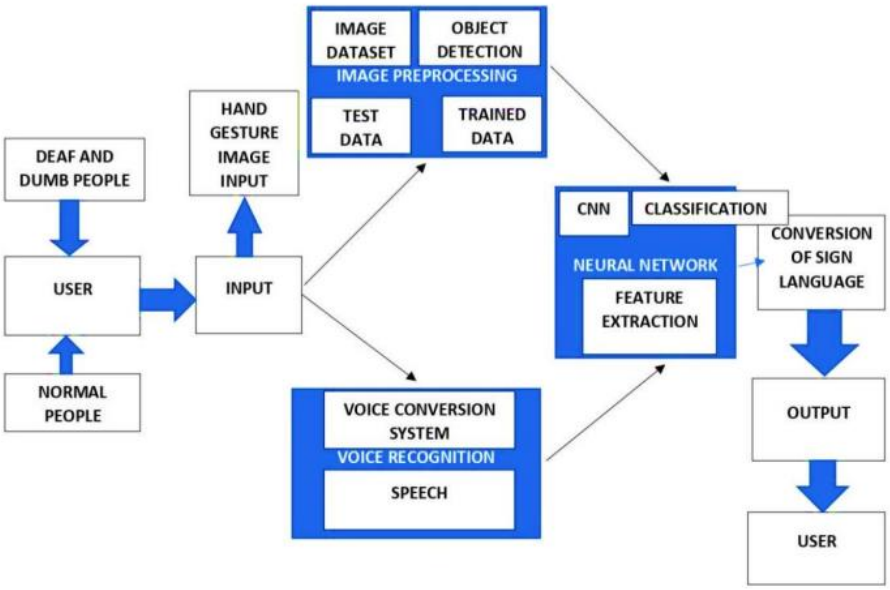
FR. NO.	**Non Functional Requirement**	**Description**
NFR - 1	Usability	User can easily upload the image and this app is designed in a way where user can easily find some predefined templates or layouts
NFR -2	Security	User should sign in into the application. So any unauthorized access will be avoided at the most.
NFR - 3	Reliability	This application has robust fault tolerance and even if there is an error, it recuperates swiftly.
NFR - 4	Performance	Utilizing the CNN model, the gestures made by the user is predicted by the application with a higher accuracy
NFR - 5	Availability	This application is effortless and is accessible to all users. The predefined formats or layouts are understandable and makes it easier for accessing it.
NFR - 6	Scalability	Highly scalable which uses gesture recognition and is a hands on model which is used in many other applications.

5. PROJECT DESIGN

5.1 Data Flow Diaaram

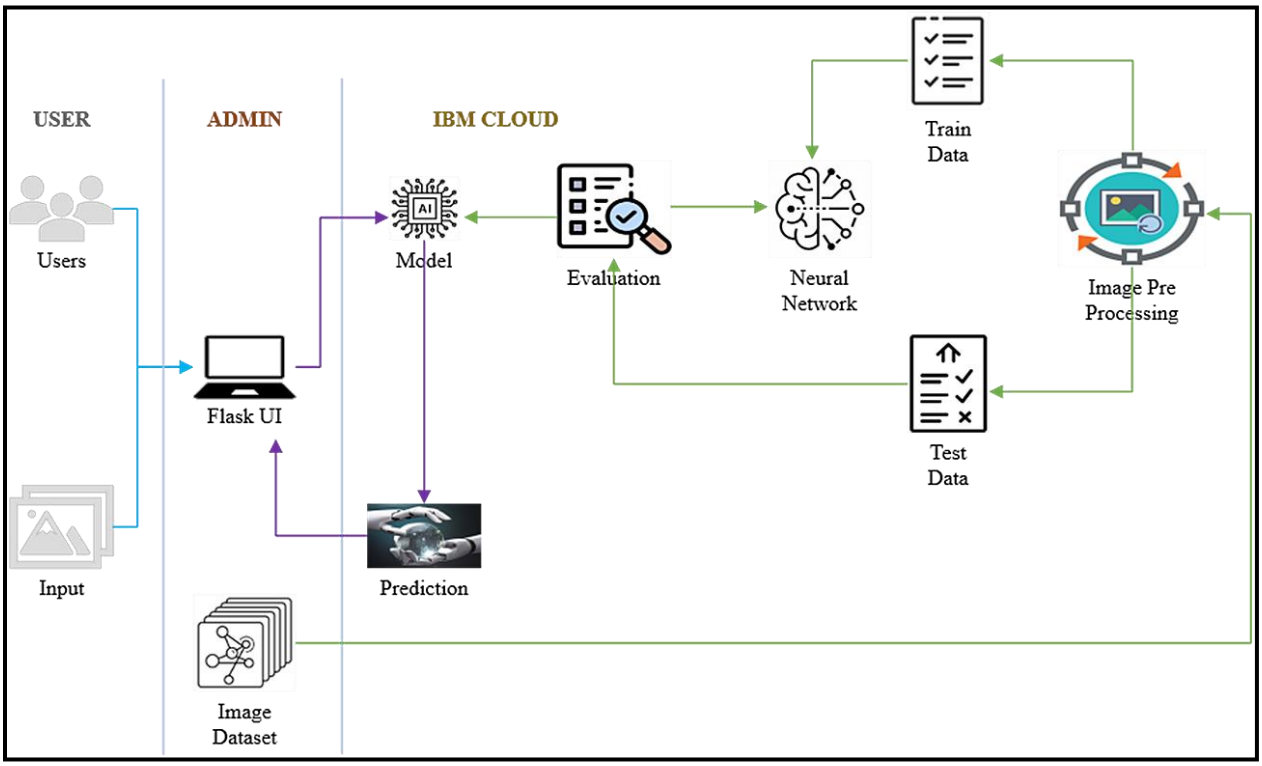


5.2 Solution and Technical Architecture



Hardware/Software Requirement

Operating System	**Windows, Mac, Linux**
WebCam	**Integrated or External with FullHD Support**
Python	**v3.9.0 or Above**
Python Packages	**Flask, Tensorflow, Opencv-Python, Keras, Numpy, Pandas**
Web Browser	**Mozilla Firefox, Google Chrome or any modern web browser**
IBM Cloud (for training)	**Watson Studio - Model Training & Deployment as Machine Learning Instance**



Components & Technologies:

SI. No.	Components	Description	Technology
1.	User Interface	The user interface is the point of human computer interaction and communication in device.	Python flask, HTML,CSS/JavaScript.
2.	Flash UI	Flash's user interface components let you interact with the users that use your site and gatherinformation.	Using the cloud,it can be executed.
3.	Models	Support Vector Machine(SVM) is subsequently applied to classify our gesture imagedataset.	Machine Learning.
4.	Image	Image processing is used to extract signsfrom the imageusing neural network.	ANN, CNN, OpenCV.
5.	Evaluatedata	Aims to estimate the generalization accuracy of a modelon future (unseen/out-of-sample) data.	NLP.
6.	Unstructured data	Unstructured data is a conglomeration of many variedtypes of data thatare stored in theirnative formats.	Natural Language Processing (NLP).
7.	Structured data	Typically categorized as quantitative data is highlyorganized and easilydecipherable by machine learning algorithms.	Machine language and artificial intelligence tools.
8.	File Storage	File storage requirements to storethe trained modelin order to use it whenever it is needed.	IBM Block Storage or Cloud object.
9.	ML service	Provides a full range of tools and services so that you can build,train, and deploy Machine Learning models.	Python, IBM Watson.
10.	IBM Cloud	IBM Watson Studioempowers data scientists, developers and analysts to build, runand manage AI models, and optimize decisions anywhere on IBM CloudPak for Data	IBM Cloud and Watson Studioservice

11.	**Dataset**	**First prototype of this systemused a dataset of 24 static signsfrom the Panamanian Manual Alphabet.**	**AI technology.**

Application Characteristics:

Si. No.	Characteristics	Description	Technology
1.	Open-Source Frameworks	Helps you implement bestpractices for dataautomation, model tracking, performance monitoring, and model retraining.	TensorFlow.
2.	SecurityImplementations	It operates the largest national network of professional monitoring centres andoffers a six-month, money-back guarantee to customers.	ADT type of coding.
3.	Scalable Architecture	Three-tier architecture is a wellestablished software application architecture that organizes applications into three logical and physical computing tiers: the datatier, the presentation tier and the application tier.	3 – TierArchitecture.
4.	Availability	The systemwill be madeubiquitous so thatit is available everywhere.	Web Application.
5.	Performance	The model will be fine-tuned to strike a balance betweenaccuracy vs performance.	Optimization of codeand trained model.

5.3 User Stories

User Type	Functional Requirement(Epic)	User Story Number	User Story/Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password	I can access my account/dashboard	High	Sprint-1
	Authentication	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Low	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password	I am able to get into the Dashboard	High	Sprint-2
	Dashboard	USN-4	One place to explore all available features	I can access my dashboard	High	Sprint-2
Customer (Web user)	Registeration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password			
	Authentication	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	Low	Sprint-1

	Login	USN-3	As a user, I can log into the application by entering email & password	I am able to get into the Dashboard	Low	Sprint- 2
	Dashboard	USN-4	One place to explore all available features	I can access my dashboard	Low	Sprint-2
	Upload image	USN-5	As a user, I can upload the sign language image for translating into text format	I can be able to see the appropriate text for the sign language	High	Sprint - 3
Administrator	Manage	USN-6	Do-it-yourself Everything. service for delivering Everything	Set of predefined requirements that must be met to mark a user story complete	High	Sprint - 4

6.PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning and Estimation

Sprint	**Functional Requirement (Epic)**	**Use Story Number**	**User Story/Task**	**Story Points**	**Priority**	**Team Members**
Sprint– 1	Registration	USN– 1	As a user, I can register for the application by entering my email,password,and confirming my password.	3	High	SASIDHARAN N
Sprint– 1	Authentication	USN– 2	As a user,I will receive OTP to confirm details.	2	High	SUJAN S

Sprint– 1	Registration	USN– 3	As a user,I will receive confirmation email onc e I have registered for the application.	1	Low	SIVANADHAN
Sprint– 1	Login	USN– 4	As a user,I can login to the application by enteri ng email & password.	2	High	VISHVA B
Sprint– 2	Dashboard	USN– 5	As a user,I must have one place to explore all ava ilable features.	3	High	SASIDHARAN N
Sprint– 2	Login	USN– 6	As a user, If I forgetmy password, I must get an auto- generated password to reset my password.	2	Medium	SIVASURYA
Sprint– 3	Help	USN– 7	As a user,I must be able to reach out to the Suppor t Team to get my issues resolved.	1	Low	VISHVA B
Sprint– 3	Management	USN– 8	As a user, I can access the site using mobile/desk top.	3	High	SUJAN S
Sprint– 4	System	USN– 9	As a user, I must have access to previous usage hi story.	2	Medium	VISHVA B
Sprint– 4	System	USN– 10	As a user, I can have audio output as well as text output.	3	High	SIVANADHAN

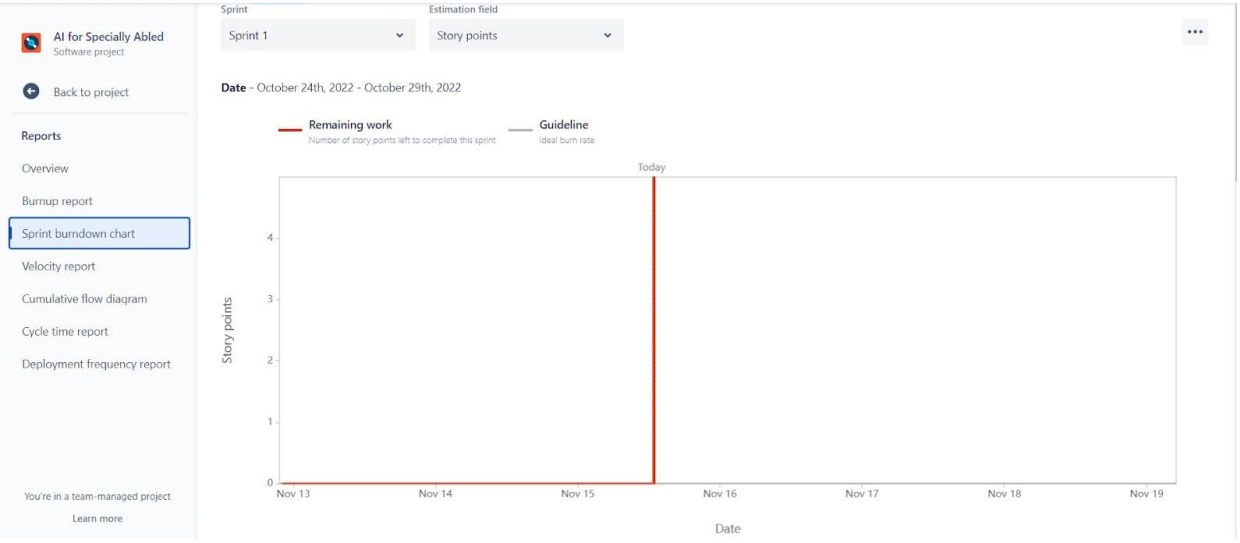
6.2 Sprint Delivery and Schedule

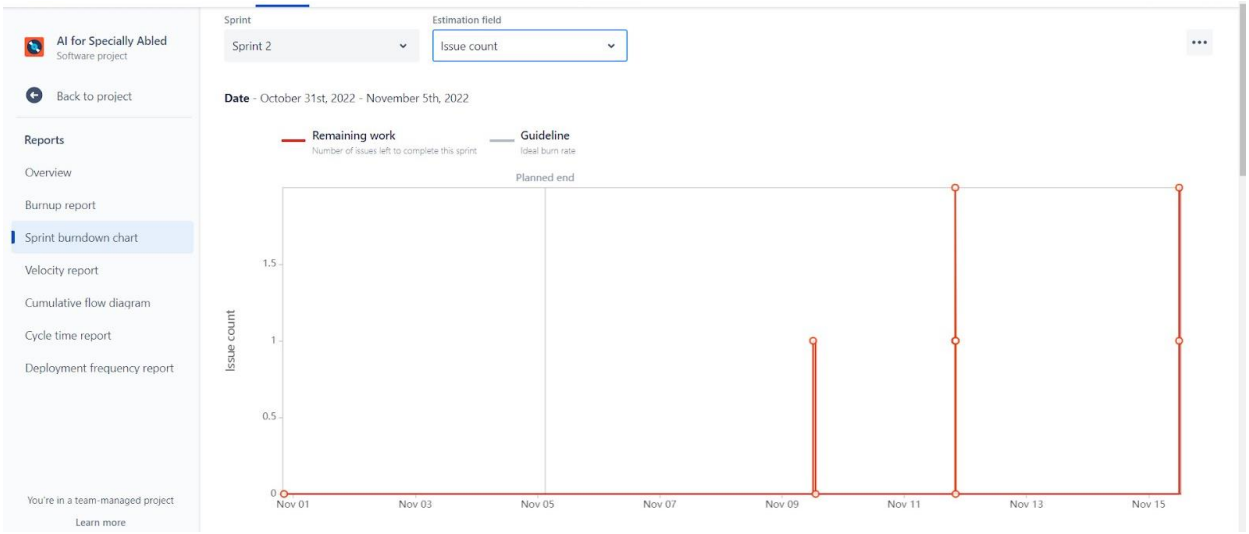
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint– 1	8	6Days	24October,2022	29October,2022		01November,2022
Sprint– 2	5	6Days	31October,2022	05November,2022		07 November,2022
Sprint– 3	4	6Days	07November,2022	12November,2022		
Sprint– 4	5	6Days	14November,2022	19November,2022		

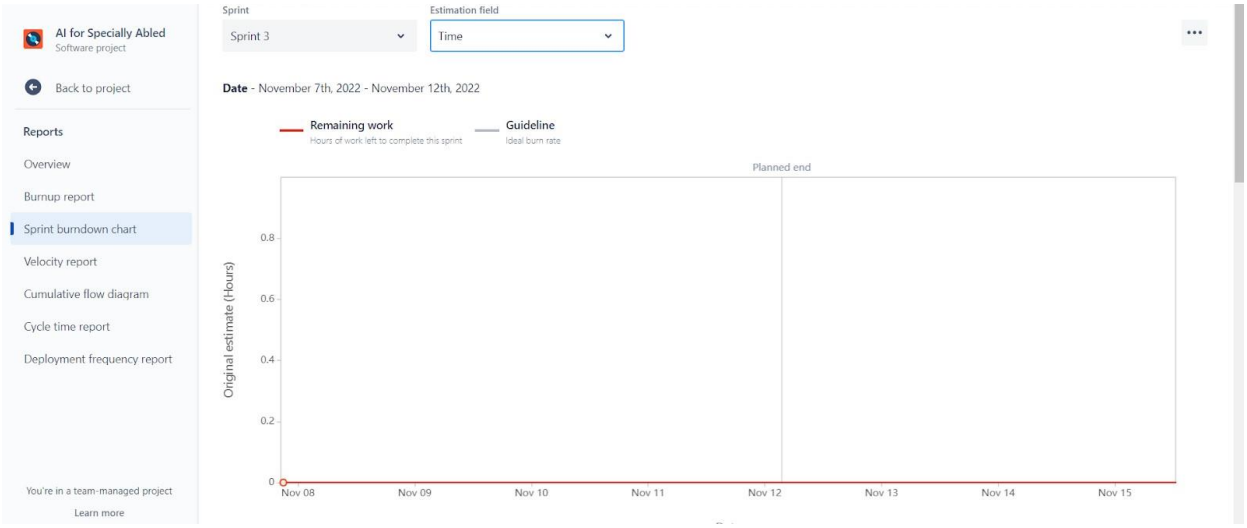
6.2 Reports from JIRA

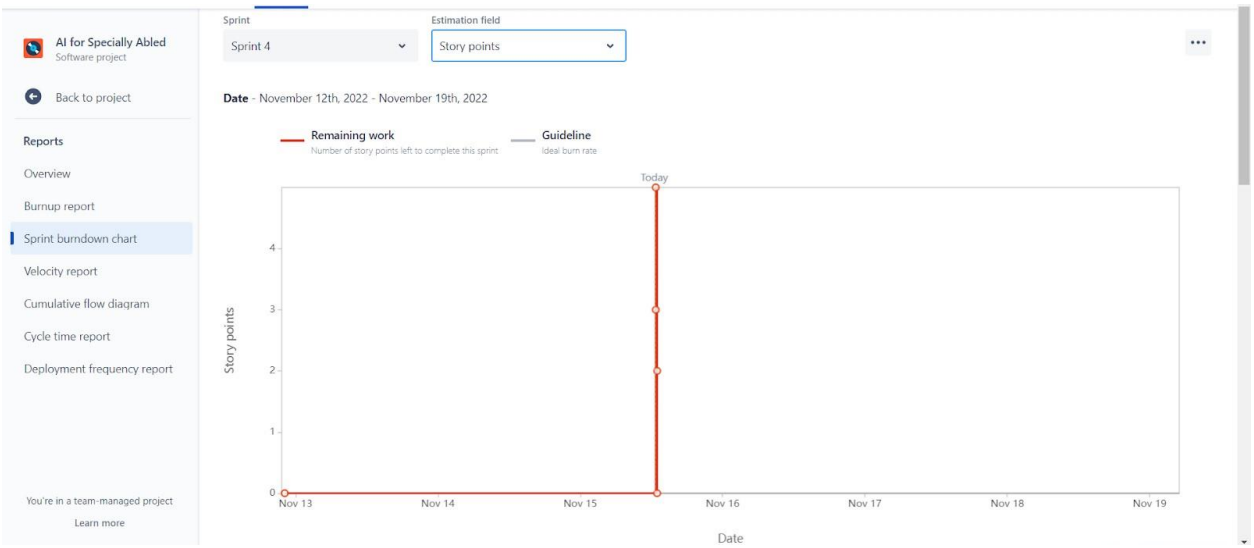
Burndown Chart

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.




AI for Specially Abled
Software project
Back to project
Reports
Overview
Burnup report
Sprint burndown chart
Velocity report
Cumulative flow diagram
Cycle time report
Deployment frequency report
You're in a team-managed project
Learn more
Sprint
Sprint 2
Estimation field
Issue count
Date - October 31st, 2022 - November 5th, 2022
Remaining work
Number of issues left to complete this sprint
Guideline
Ideal burn rate
Planned end
Issue count
0
0.5
1
1.5
Nov 01
Nov 03
Nov 05
Nov 07
Nov 09
Nov 11
Nov 13
Nov 15


AI for Specially Abled
Software project
Back to project
Reports
Overview
Burnup report
Sprint burndown chart
Velocity report
Cumulative flow diagram
Cycle time report
Deployment frequency report
You're in a team-managed project
Learn more
Sprint
Sprint 3
Estimation field
Time
Date - November 7th, 2022 - November 12th, 2022
Remaining work
Hours of work left to complete this sprint
Guideline
Ideal burn rate
Planned end
Original estimate (Hours)
0
0.2
0.4
0.6
0.8
Nov 08
Nov 09
Nov 10
Nov 11
Nov 12
Nov 13
Nov 14
Nov 15



Velocity Report

Velocity is the average amount of work a scrum team completes during a sprint. In teammanaged Jira Software projects, this can be measured in either story points or number of issues.

7.CODING AND SOLUTIONING

7.1 Source Code

In this, we will pre-process the images which will be used for building the model. Image preprocessing includes zooming, shearing, flipping to increase the robustness of the model after it is built. We will be using the Keras package for pre-processing images.

Import ImageDataGenerator and create an instance for which include shearing, rescale, zooming, etc to make the model robust with different types of images.

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

[1]

Python

We will be loading all the images of the train and test using the flow from directory method.

```
x_train = train_datagen.flow_from_directory('/content/Dataset/training_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
[12]                                                                                                                                                                    Python
...  Found 15750 images belonging to 9 classes.

x_test = test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")
[13]                                                                                                                                                                    Python
...  Found 2250 images belonging to 9 classes.
```

We start building our model by:

1.Initializing the model

2. Adding Convolution layers
3. Adding Pooling layers
4. Flatten layer
5. Full connection layers which include hidden layers

At last, we compile the model with layers we added to complete the neural network structure. Import the libraries that are required to initialize the neural network layer, create and add different layers to the neural network model.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
[6]                                                                                                                                                                     Python
```

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

```
model = Sequential()
[8]                                                                                                                                                                     Python
```

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).

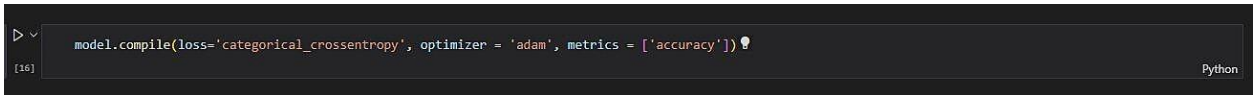
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,1), activation='relu'))
#no. of feature detectors, size of feature detector, image size, activation function
[11]                                                                                                                                   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

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.

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.

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

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.



Fit the neural network model with the train and test set, number of epochs, and validation steps. The weights are to be saved for future use. The weights are saved in as .h5 file using save().

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

[17] Python

```
/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.

Epoch 1/10
24/24 [==============================] - ETA: 0s - loss: 1.0716 - accuracy: 0.7176

WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 40 batches). You may need to use the repeat() function when building your dataset.

24/24 [==============================] - 96s 4s/step - loss: 1.0716 - accuracy: 0.7176 - val_loss: 0.4701 - val_accuracy: 0.9107
Epoch 2/10
24/24 [==============================] - 82s 3s/step - loss: 0.2010 - accuracy: 0.9400
Epoch 3/10
24/24 [==============================] - 94s 4s/step - loss: 0.0867 - accuracy: 0.9751
Epoch 4/10
24/24 [==============================] - 85s 4s/step - loss: 0.0403 - accuracy: 0.9893
Epoch 5/10
24/24 [==============================] - 82s 3s/step - loss: 0.0289 - accuracy: 0.9915
Epoch 6/10
24/24 [==============================] - 82s 3s/step - loss: 0.0209 - accuracy: 0.9949
Epoch 7/10
24/24 [==============================] - 83s 3s/step - loss: 0.0137 - accuracy: 0.9957
Epoch 8/10
24/24 [==============================] - 81s 3s/step - loss: 0.0090 - accuracy: 0.9979
Epoch 9/10
24/24 [==============================] - 82s 3s/step - loss: 0.0153 - accuracy: 0.9957
Epoch 10/10
24/24 [==============================] - 81s 3s/step - loss: 0.0086 - accuracy: 0.9986

<keras.callbacks.History at 0x7fc0ea424290>
```



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

Python

Now we test the model by passing an image to get predictions. While test the model we should make sure that the test image should meet the target size of the model, dimensions need to meet, and should undergo rescaling before giving it to the model.

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

[17] Python

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

[18] Python

```python
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 = np.argmax(prediction,axis=1)
    print(prediction)
```

Python

```python
frame=cv2.imread('/content/Dataset/test_set/G/1.png')
data = detect(frame)
```

Python

```
1/1 [==============================] - 0s 25ms/step
[[2.9662006e-09 3.0511607e-09 5.7518361e-07 2.6636766e-09 7.6029876e-09
  1.4324395e-08 9.9982303e-01 1.7639149e-04 1.6517550e-09]]
[6]
```

Now we will be building a Flask application that is used for building our UI which in backend can be interfaced to the model to get predictions. Flask application requires an HTML page for Frontend and a Python file for the backend which takes care of the interface with the model.

app.py

app.py > ...

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

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

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

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


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

camera.py

```python
import cv2
import numpy as np
from keras.models import load_model
from keras.utils import load_img, img_to_array

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('aslpng1.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 = load_img('image.jpg', target_size=(64,64))
        x = 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)
        ret,jpg = cv2.imencode('.jpg', frame)
        return jpg.tobytes()
```


main.py

```python
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()
```

index.html

templates > index.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>Sign Language Detection</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(247, 246, 244);">
    <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
                        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;borde
                <img src="{{ url_for('video_feed') }}" style="width: 100%;height: 100%;color: rgb(0, 0, 0);text-align: ce
                    alt="Camera Access Not Provided!">
            </div>
        </div>
```

7.2 IBM model training

```
In [1]: from ibm_watson_machine_learning import APIClient
        wml_credentials = {
               "url": "https://us-south.ml.cloud.ibm.com",
               "apikey": "mNVF7E95G-awR213njShj1GiUfN-1SpPq-ko8Wx7na1-"
        }

        client = APIClient(wml_credentials)

In [2]: def guid_from_space_name(client, space_name):
            space = client.spaces.get_details()
            return (next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])

In [3]: space_uid = guid_from_space_name(client, 'communication_model_deployment')
        print("Space UID : ", space_uid)

        Space UID :  21c15ae0-ee26-497d-b615-eb30ef2e16fe

In [4]: client.set.default_space(space_uid)

Out[4]: 'SUCCESS'

In [5]: client.repository.download("cefca265-2301-4620-897a-9c80d6ff7f1a","IBM_Model_Download.tar.gz")

        Successfully saved model content to file: 'IBM_Model_Download.tar.gz'
Out[5]: '/home/wsuser/work/IBM_Model_Download.tar.gz'

In [ ]:
```



```
In [1]: pwd

Out[1]: '/home/wsuser/work'

In [2]: from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

In [4]: import os, types
        import pandas as pd
        from botocore.client import Config
        import ibm_boto3

        def __iter__(self): return 0

        # @hidden_cell
        # The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
        # You might want to remove those credentials before you share the notebook.
        client_9e2ebbcc4db04f3fb5d87e8fa4800e36 = ibm_boto3.client(service_name='s3',
            ibm_api_key_id='GN91c7sTtlR1DYfMzZqIUwGSIkATCCriPItR-s81P82-',
            ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
            config=Config(signature_version='oauth'),
            endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

        streaming_body_1 = client_9e2ebbcc4db04f3fb5d87e8fa4800e36.get_object(Bucket='communicationmodeltraining-donotdelete-pr-xpzs67frbbb7s3', Key='Communication_Dataset.zip

        # Your data file was loaded into a botocore.response.StreamingBody object.
        # Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
        # ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
        # pandas documentation: http://pandas.pydata.org/
```

In [5]:
```
# Unzip the Dataset Zip File
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()), 'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [6]:
```
%%bash
ls Communication_Dataset
```

```
test_set
training_set
```

In [7]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/home/wsuser/work/Communication_Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/home/wsuser/work/Communication_Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 27000 images belonging to 9 classes.
Found 25737 images belonging to 9 classes.
```

In [8]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  30
Len x-test :  29
```

In [9]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```


Out[9]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Saving the Model

In [15]:
```
model.save('IBM_Communication_Model.h5')
# Current accuracy is 0.8154
```

In [16]:
```
# Convert the Saved Model to a Tar Compressed Format
!tar -zcvf IBM_TrainedModel.tgz IBM_Communication_Model.h5
```

```
IBM_Communication_Model.h5
```

In [17]:
```
%%bash
ls -ll
```

```
total 296080
drwxrwx--- 4 wsuser wscommon      4096 Nov  8 12:46 Communication_Dataset
-rw-rw---- 1 wsuser wscommon 111324760 Nov  8 13:17 IBM_Communication_Model.h5
-rw-rw---- 1 wsuser wscommon  96688252 Nov  8 12:42 IBM_Model_Download.tar.gz
-rw-rw---- 1 wsuser wscommon  95160335 Nov  8 13:17 IBM_TrainedModel.tgz
```

Watson Machine Learning

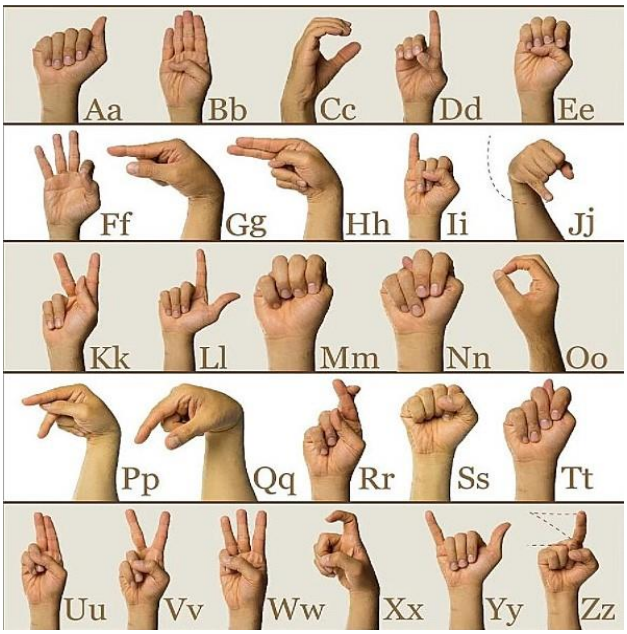
In [18]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "mNVF7E95G-awR213njShj1GiUfN-1SpPq-ko8Wx7na1-"
}

client = APIClient(wml_credentials)
```

8. TESTING

8.1 Test Cases

This project only predicts alphabets from “A” to “I” in American Sign Language



8.2 User Acceptance Testing

Defect Analysis

This report shows the number of resolved or closedbugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	0	1	0	6
Duplicate	1	0	0	0	1
External	0	2	0	0	2
Fixed	7	0	0	0	7

Not Reproduced	0	1	0	0	1
Skipped	0	0	1	0	1
Won't Fix	0	4	0	0	4
Totals	13	7	2	0	44

Testcase Analysis

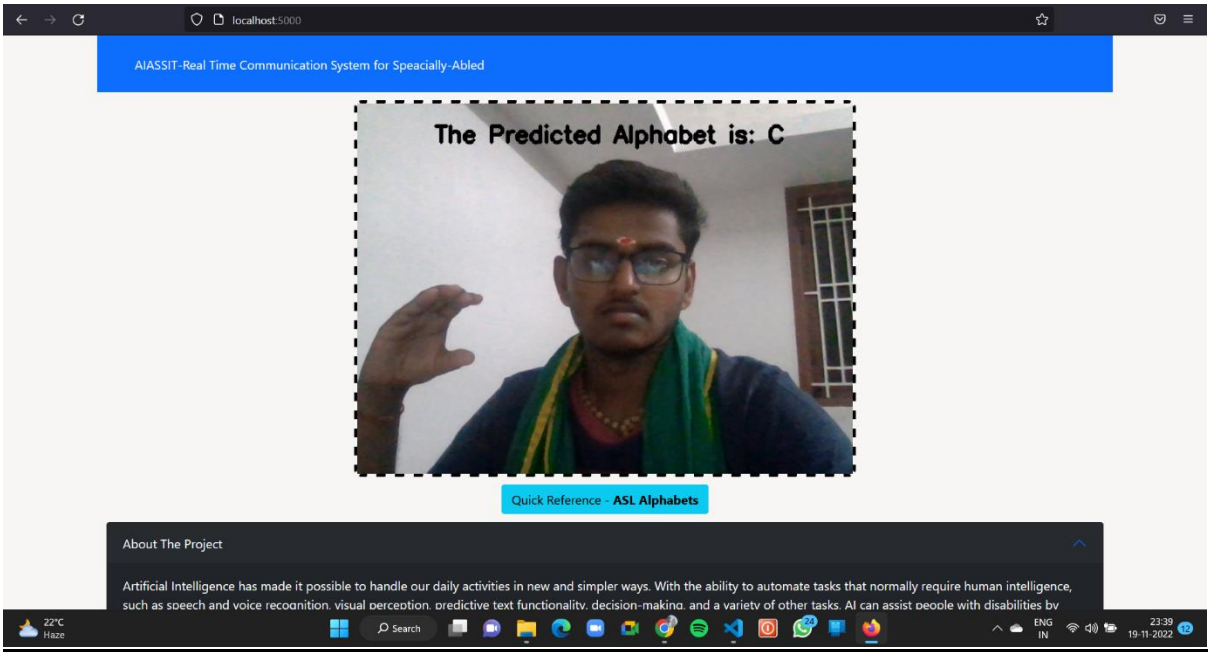
This report shows the number of test cases that have passed, failed, and untested.

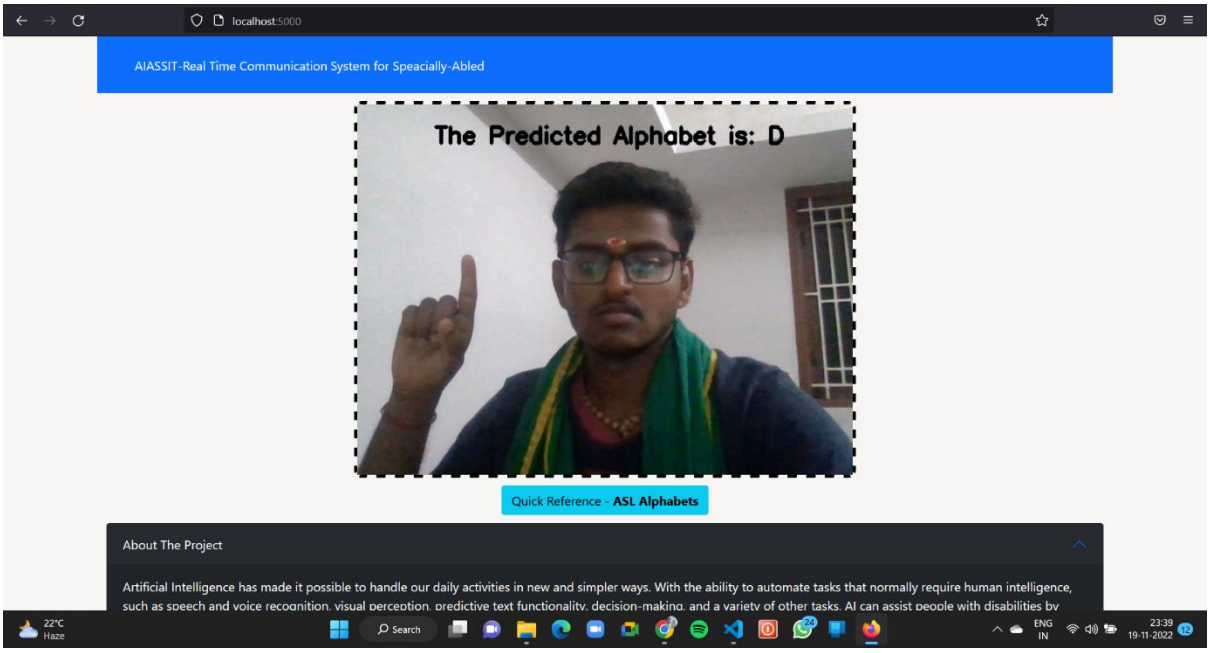
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	9	0	4	5
Client Application	9	0	4	5
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	5	0	0	5
Final Report Output	9	0	4	5
Version Control	2	0	0	2

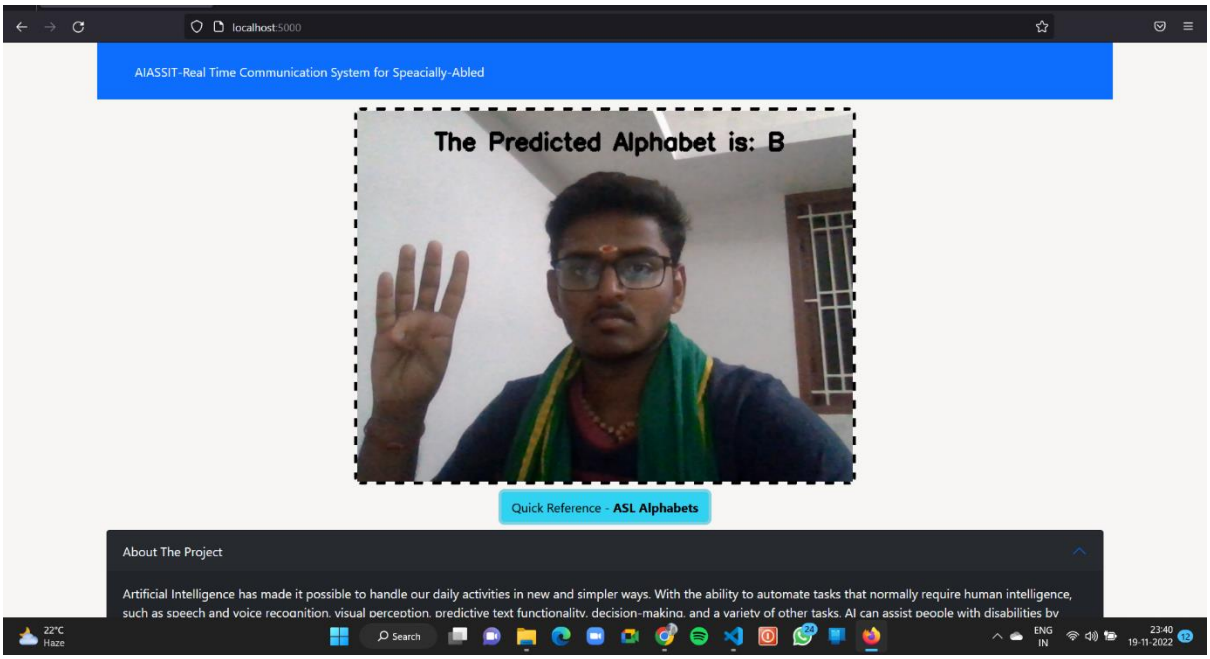
9.RESULTS

The proposed procedure was implemented and tested with set of images.
The set of 18789 images of Alphabets from “A” to “I” are used for training database and a set of 2350 images of Alphabets from “A” to “I” are used for testing database.
Once the gesture is recognize the equivalent Alphabet is shown on the screen.

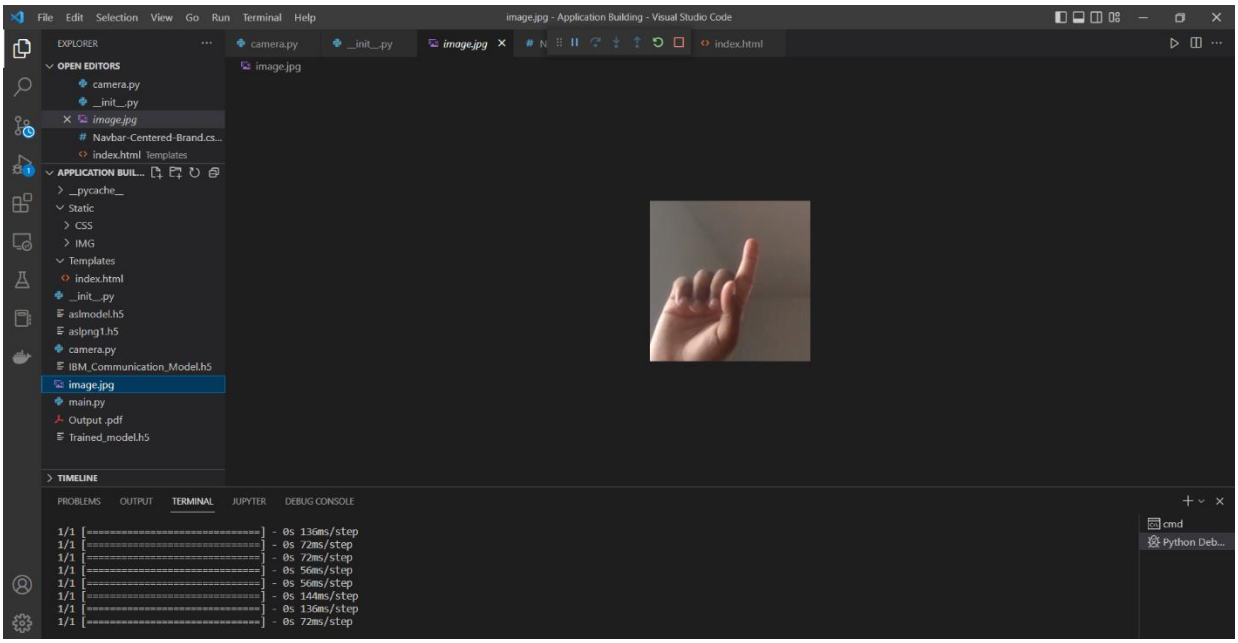
Some sample images of the output are provided below:



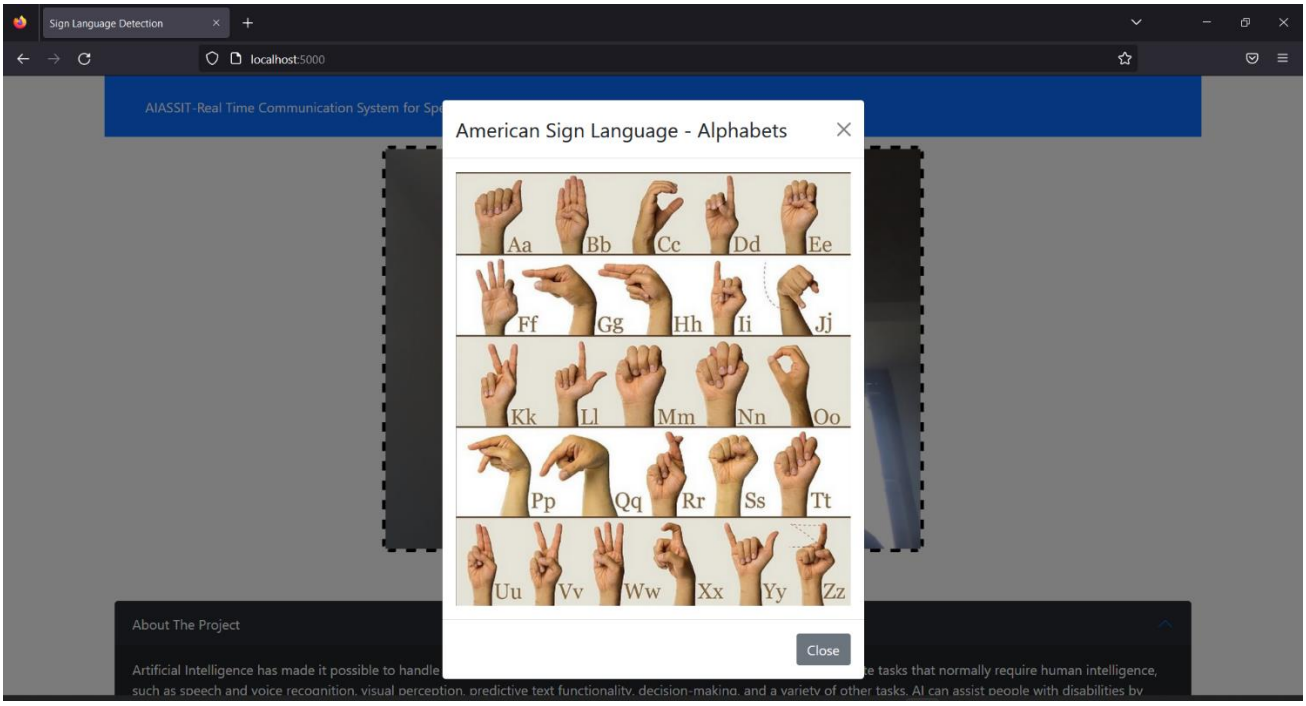




REGION OF INTEREST:



AMERICAN SIGN LANGUAGE STANDARD REFERENCE:



9.1 Performance Metric

Model Performance Testing:

Project team shall fill the following information on in model performance testing template.

S.No.	Parameter	Values
1.	Model Summary	Total params: 15,750,473 Trainable params: 15,750,473 trainable params: 0
2.	Accuracy	Training Accuracy – 99.40% Valida on Accuracy -90.44%

Screenshot:

1.Model Summary

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 512)               15745536

 dense_1 (Dense)             (None, 9)                 4617

=================================================================
Total params: 15,750,473
Trainable params: 15,750,473
Non-trainable params: 0
_________________________________________________________________
```

2.Accuracy

```
In [ ]: 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

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versio
n. Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.
Epoch 1/10
24/24 [==============================] - ETA: 0s - loss: 0.7714 - accuracy: 0.7442
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch *
epochs` batches (in this case, 40 batches). You may need to use the repeat() function when building your dataset.
24/24 [==============================] - 2691s 113s/step - loss: 0.7714 - accuracy: 0.7442 - val_loss: 0.3322 - val_accuracy: 0.9031
Epoch 2/10
24/24 [==============================] - 986s 41s/step - loss: 0.1882 - accuracy: 0.9488
Epoch 3/10
24/24 [==============================] - 521s 22s/step - loss: 0.0964 - accuracy: 0.9745
Epoch 4/10
24/24 [==============================] - 287s 12s/step - loss: 0.0617 - accuracy: 0.9819
Epoch 5/10
24/24 [==============================] - 154s 6s/step - loss: 0.0464 - accuracy: 0.9868
Epoch 6/10
24/24 [==============================] - 92s 4s/step - loss: 0.0332 - accuracy: 0.9928
Epoch 7/10
24/24 [==============================] - 63s 3s/step - loss: 0.0251 - accuracy: 0.9948
Epoch 8/10
24/24 [==============================] - 50s 2s/step - loss: 0.0176 - accuracy: 0.9964
Epoch 9/10
24/24 [==============================] - 42s 2s/step - loss: 0.0134 - accuracy: 0.9973
Epoch 10/10
24/24 [==============================] - 44s 2s/step - loss: 0.0143 - accuracy: 0.9967
Out[ ]:

In [ ]: model.save('aslpng1.h5')
```

10.ADVANTAGES & DISADVANTAGES

Advantages:

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

2. As different language exist, their dataset can be added, and the user can choose which language to listen the message from specially abled person.

Disadvantages:

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

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

3. In absence of gesture recognition, alphabets from J cannot be identified as they require some kind of gesture input from the user.

11.CONCLUSION

➤ Sign language is a useful tool for facilitating communication between deaf and hearing people. Because it allows for two-way communication, the system aims to bridge the communication gap between deaf people and the rest of society.

➤ This system sends hand gestures to the model, who recognizes them and displays the equivalent Alphabet on the screen.

➤ Deaf-mute people can use their hands to perform sign language, which will then be converted into alphabets, thanks to this project.

➤ The 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.

12.FUTURE SCOPE

➤ 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 the specially abled people such as deaf and dumb.

➤ With 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.

13.ANNEXURE

Github link: https://github.com/IBM-EPBL/IBM-Project-28010-1660105579

Demo link : https://drive.google.com/file/d/1QHrfifLHsg5UByyFytjX-GCga25Kqdrl/view?usp=share_link

Source code:

MODEL BUILDING AND TESTING:

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

x_train = train_datagen.flow_from_directory('/content/Dataset/training_set',target_si
ze=(64,64),batch_size=300,class_mode='categorical',color_mode="grayscale")

x_test = test_datagen.flow_from_directory('/content/Dataset/test_set',target_size=(6
4,64),batch_size=300,class_mode='categorical',color_mode="grayscale")

from keras.models import Sequential from keras.layers
import Dense
```

```
from keras.layers import Convolution2D from
keras.layers import MaxPooling2D from keras.layers
import Dropout from keras.layers import Flatten

model = Sequential()

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

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

model.add(Flatten())

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

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

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

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

model.save('aslpng1.h5')

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

model=load_model('aslpng1.h5')


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 = np.argmax(prediction,axis=1)
 print(prediction)

frame=cv2.imread('/content/Dataset/test_set/G/1.png') data = detect(frame[0])
```

HTML CODE:

```
<!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>Sign Language Detection</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(247, 246, 244);">
   <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 dflex justify-content-center align-items-center
me-2 bs-icon"><i
                class="fas fa-flask"></i></span><span style="color: rgb(255,255,255);">Real-Time
Communication
            System Powered By AI&amp;nbsp;For Specially
```

```
Abled</span></a>
        <div></div>
      </div>
  </nav>
  <section>
    <div class="d-flex flex-column justify-content-center align-itemscenter">
      <div class="d-flex flex-column justify-content-center alignitems-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(0, 0, 0) ;">
        <img src="{{ url_for('video_feed') }}" style="width:
100%;height: 100%;color: rgb(0, 0, 0);text-align: center;font-size: 20px;"
          alt="Camera Access Not Provided!">
      </div>
    </div>
    <div class="d-flex flex-column justify-content-center align-itemscenter" 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" ariaexpanded="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>
    </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" arialabel="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 btnsecondary"
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>
```

FLASK APPLICATION:

```
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()
```

USING CAMERA : import

```
cv2 import numpy as np
from keras.models import load_model from keras.utils import
load_img, img_to_array

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('aslpng1.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 =
                    load_img('image.jpg', target_size=(64,64)) x = 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()
```

MAIN FUNCTION:

```
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()
```

CSS CODE:

Banner-Heading-Image.css

```
.fit-cover {
  object-fit: cover;
}
```

Navbar-Centered-Brand.css

```
.bs-icon {
  --bs-icon-size: .75rem;
```

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

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

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

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

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

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

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

```
  background: var(--bs-primary);
}

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

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

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

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

IBM MODEL TRAINING:

```
from ibm_watson_machine_learning import APIClient wml_credentials = {
            "url": "https://us-south.ml.cloud.ibm.com",
                                    "apikey": "mNVF7E95G-awR213njShj1GiUfN-1SpPq-ko8Wx7na1-"
}
client = APIClient(wml_credentials)

def guid_from_space_name(client, space_name):
   space = client.spaces.get_details()
   return (next(item for item in space['resources'] if item['entity']["name"]
== space_name)['metadata']['id'])

space_uid = guid_from_space_name(client, 'communication_model_deployment') print("Space UID : ",

space_uid) client.set.default_space(space_uid)
```

```
client.repository.download("cefca265-2301-4620-897a-
9c80d6ff7f1a","IBM_Model_Download.tar.gz") from tensorflow.keras.preprocessing.image import

ImageDataGenerator

# Training Datagen train_datagen =
ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_fli p=False) # Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)

import os, types import pandas as pd from
botocore.client import Config import ibm_boto3
def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials. # You might want
to remove those credentials before you share the notebook. client_9e2ebbcc4db04f3fb5d87e8fa4800e36 =
ibm_boto3.client(service_name='s3',
    ibm_api_key_id='GN91c7sTtlR1DYfMzZqIUwGSIkATCCriPItR-s81P82-',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

streaming_body_1 = client_9e2ebbcc4db04f3fb5d87e8fa4800e36.get_object(Bucket='communicationmodeltr aining-donotdelete-
pr-xpzs67frbbb7s3', Key='Communication_Dataset.zip')['Body'] # Your data file was loaded into a
botocore.response.StreamingBody object. # Please read the documentation of ibm_boto3 and pandas to learn more about the
possibilities to load the data. # ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/

# Unzip the Dataset Zip File from io import BytesIO import zipfile unzip =
zipfile.ZipFile(BytesIO(streaming_body_1.read()), 'r') file_paths = unzip.namelist() for path in
file_paths:
    unzip.extract(path)

%%bash ls Communication_Dataset

# Training Dataset x_train=train_datagen.flow_from_directory(r'/home/wsuser/work/Communication
_Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/home/wsuser/work/Communication_D
ataset/test_set',target_size=(64,64),
class_mode='categorical',batch_size=900)

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

# The Class Indices in Training Dataset x_train.class_indices
```

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

# Creating Model
model=Sequential() #
Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D(pool_size=(2,2))) model.add(Flatten())
# Adding Hidden Layers
model.add(Dense(300,activation='relu')) model.add(Dense(150,activation='relu'))

# Adding Output Layer model.add(Dense(9,activation='softmax'))

# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['a ccuracy'])

# Fitting the Model Generator
model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validati
on_data=x_test,validation_steps=len(x_test))

model.save('IBM_Communication_Model.h5')
# Current accuracy is 0.8154

# Convert the Saved Model to a Tar Compressed Format
!tar -zcvf IBM_TrainedModel.tgz IBM_Communication_Model.h5

%%bash ls -
ll


from ibm_watson_machine_learning import APIClient wml_credentials = {
   "url": "https://us-south.ml.cloud.ibm.com",
   "apikey": "mNVF7E95G-awR213njShj1GiUfN-1SpPq-ko8Wx7na1-" } client =

APIClient(wml_credentials)

def guid_from_space_name(client, space_name):
   space = client.spaces.get_details()
   return (next(item for item in space['resources'] if
item['entity']["name"] == space_name)['metadata']['id'])
```

space_uid = guid_from_space_name(client, 'communication_model_deployment') print("Space UID : ", space_uid)

client.set.default_space(space_uid) client.software_specifications.list()

software_spec_uid =
client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9") software_spec_uid

```
model_details = client.repository.store_model(model='IBM_TrainedModel.tgz', meta_props={
   client.repository.ModelMetaNames.NAME: "CNN",
   client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid,
   client.repository.ModelMetaNames.TYPE: "tensorflow_2.7"}) model_id =
client.repository.get_model_uid(model_details)
```

model_id