



V.S.B. ENGINEERING COLLEGE

(Affiliated by Anna University)

Karudayampalayam, Karur-639111



Real-Time Communication System Powered by AI for Specially Abled person

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

**BACHELOR OF ENGINEERING
IN**

COMPUTER SCIENCE AND ENGINEERING

Submitted by

TEAM ID: PNT2022TMID33352

TEAM LEADER : KAVIPRIYA G (922519104072)

TEAM MEMBER 1: MOHANAPRIYA C (922519104094)

TEAM MEMBER 2: MOULIKA P (922519104096)

TEAM MEMBER 3: KEERTHANA P T (922519104078)

CONTENTS

1. **INTRODUCTION**
 - a. Project Overview
 - b. Purpose
2. **LITERATURE SURVEY**
 - a. Existing problem
 - b. References
 - c. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
 - a. Empathy Map Canvas
 - b. Ideation & Brainstorming
 - c. Proposed Solution
 - d. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
 - a. Functional requirement
 - b. Non-Functional requirements
5. **PROJECT DESIGN**
 - a. Data Flow Diagrams
 - b. Solution & Technical Architecture
 - c. User Stories
6. **PROJECT PLANNING & SCHEDULING**
 - a. Sprint Planning & Estimation
 - b. Sprint Delivery Schedule
 - c. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
 - a. Model Building
8. **TESTING**
 - a. Test Cases
 - b. User Acceptance Testing
9. **RESULTS**
 - a. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION**
12. **FUTURE SCOPE**
13. **APPENDIX**

Source Code & GitHub Link

ABSTRACT:

In our society, we have people with disabilities. The technology is developing day by day but no significant developments are undertaken for the betterment of these people. Communications between deaf-mute and a normal person has always been a challenging task. It is very difficult for mute people to convey their message to normal people. Since normal people are not trained on hand sign language. In emergency times conveying their message is very difficult. The human hand has remained a popular choice to convey information in situations where other forms like speech cannot be used. Voice Conversion System with Hand Gesture Recognition and translation will be very useful to have a proper conversation between a normal person and an impaired person in any language. New system that helps convert sign language to text and speech for easier communication with audience.

1.INTRODUCTION:

a. Project Overview:

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

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

b. Purpose:

The project aims to develop system that converts the sign language into a human hearing voice in the desired language to convey a message to normal people, as well as convert speech into understandable sign language for the deaf and dumb using unconventional neural network.

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

2. LITERATURE SURVEY:

a. Existing problem:

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

b. References:

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

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

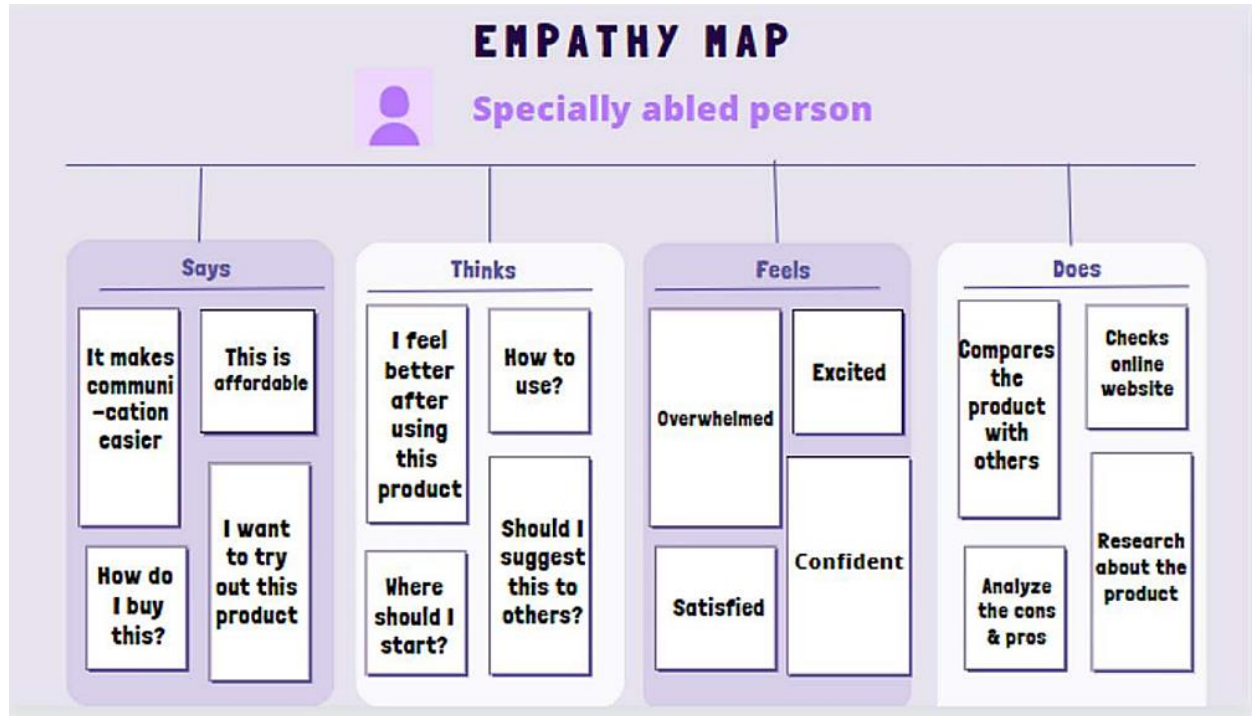
c. PROBLEM STATEMENT DEFINITION:

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

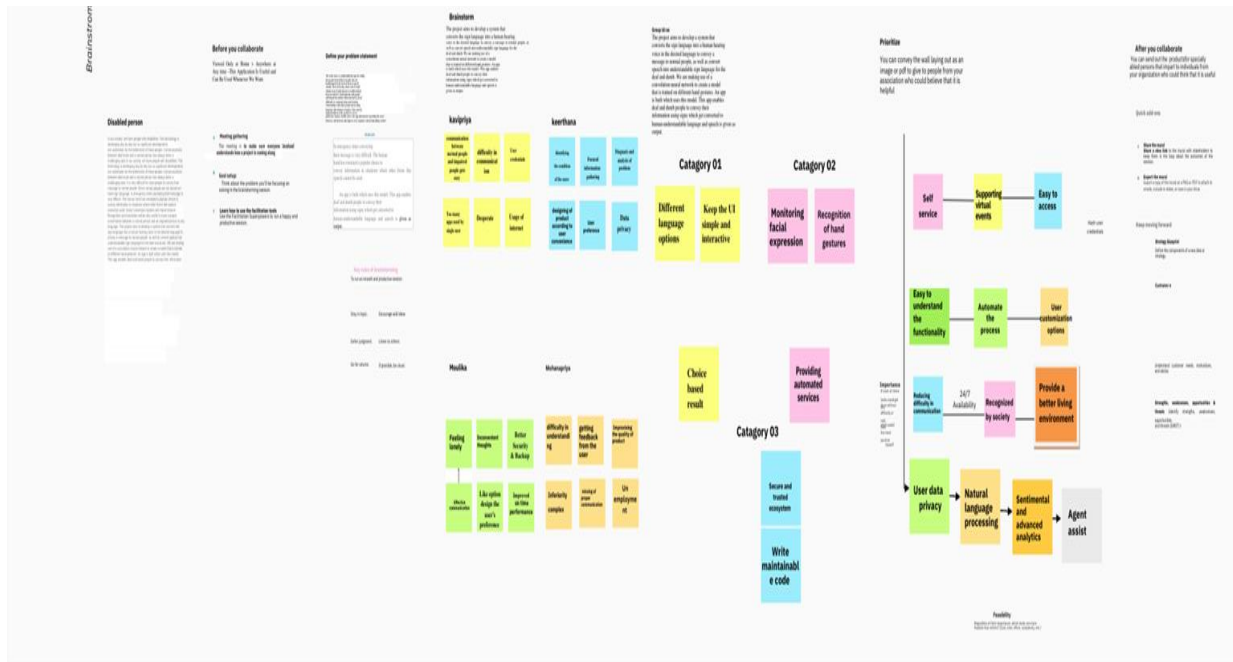
Communications between deaf-mute and a normal person has always been a challenging task. It is very difficult for mute people to convey their message to normal people. Since normal people are not trained on hand sign language.

3. IDEATION & PROPOSED SOLUTION:

a. Empathy Map Canvas:



b. Ideation & Brainstorming:



c. Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<p>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.</p>
2.	Idea / Solution description	<p>The project aims to develop a system that converts the sign language into a human hearing voice in the desired language to convey a message to normal people, as well as convert speech into understandable sign language for the deaf and dumb. We are making use of a convolution neural network to create a model that is trained on different hand gestures. An app is built which uses this model. This app enables deaf and dumb people to convey their information using signs which get converted to human-understandable language and speech is given as output.</p>

d. Problem Solution fit:

Problem-Solution fit canvas 2.0

Purpose / Vision	
1. CUSTOMER SEGMENT(S) Specially abled person	6. CUSTOMER CONSTRAINTS spending power, budget, no cash, network connection, available devices.
5. AVAILABLE SOLUTIONS In this application, we provide feedback pop-ups frequently and an emergency ping for people who have minimum knowledge about the application.	
2. JOBS-TO-BE-DONE / PROBLEMS Concentrate on making their communication much easier and live a normal life.	9. PROBLEM ROOT CAUSE Mostly genetic problems and some problems may be caused because of some viral infections. Poverty and malnutrition, accidents, poor health care.
7. BEHAVIOUR The customer will be provided with a customer care number and also there will be many feedback pop-ups frequently which helps the customer to contact us and get their jobs done.	
3. TRIGGERS Advertising the product in specially abled schools and create awareness about this product the help of government. And also advertise using various online platforms such as you tube, face book etc.	10. YOUR SOLUTION Facial recognition, Voice recognition and predictive texting tools allows people who have difficulties in speaking to communicate more easily using Artificial Intelligence. We can also use AI Sensors to monitor their health conditions regularly and save the health reports for future purposes in a separate database.
4. EMOTIONS: BEFORE / AFTER Before: Before using this product specially abled people struggled to communicate with others. After: But after using this product they will feel easy and comfortable to communicate. They feel more confident about themselves.	8. CHANNELS of BEHAVIOUR 8.1 ONLINE So, in online the customer will use various online voice assistant technologies such as Siri, Alexa, Google Assistants etc. 8.2 OFFLINE Connecting with people might be difficult depending on the type of disabilities. So, technology and AI leaves no one behind and can benefit persons with impairments.



Problem-Solution fit canvas is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 license Created by Daria Nepriakhina / Amaltama.com



4. REQUIREMENT ANALYSIS:

a. Functional requirement:

Functional Requirements:

Following are the functional requirements of the proposed solution.

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

B. Non-Functional requirements:

Non-functional Requirements:

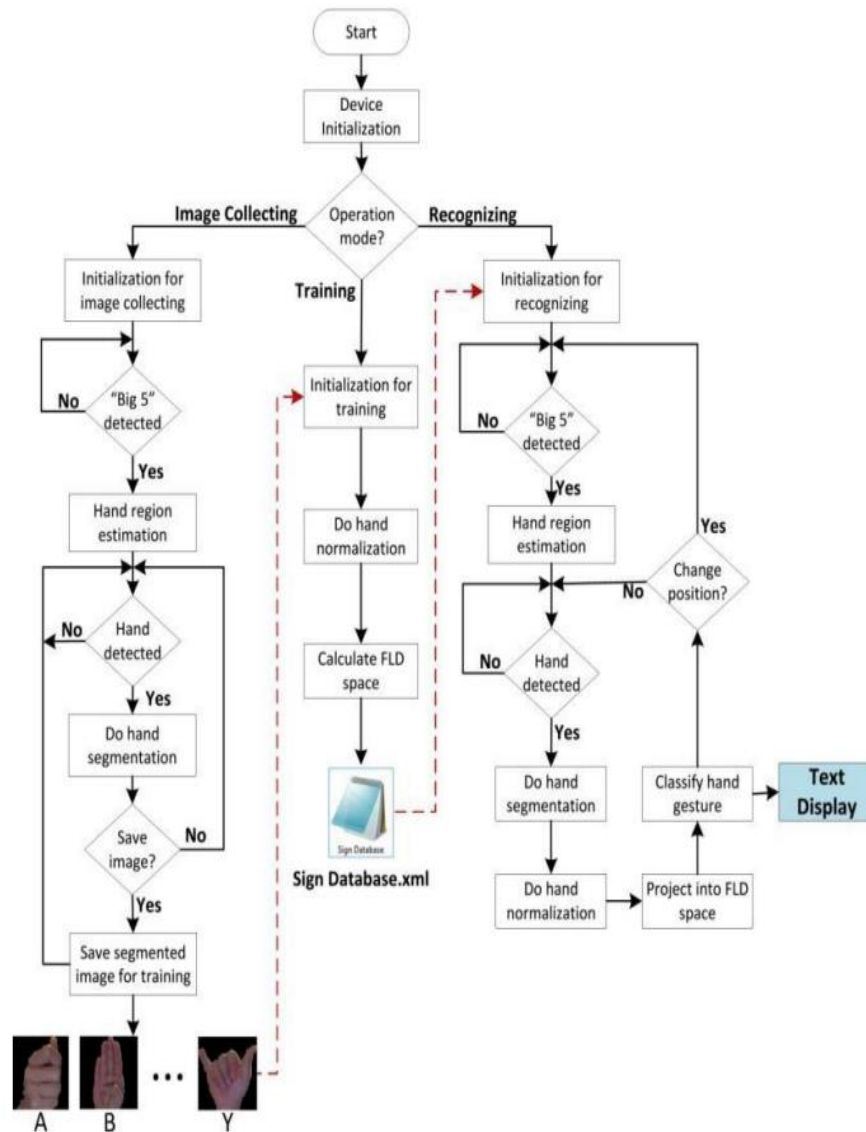
Following are the non-functional requirements of the proposed solution.

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

5. PROJECT DESIGN:

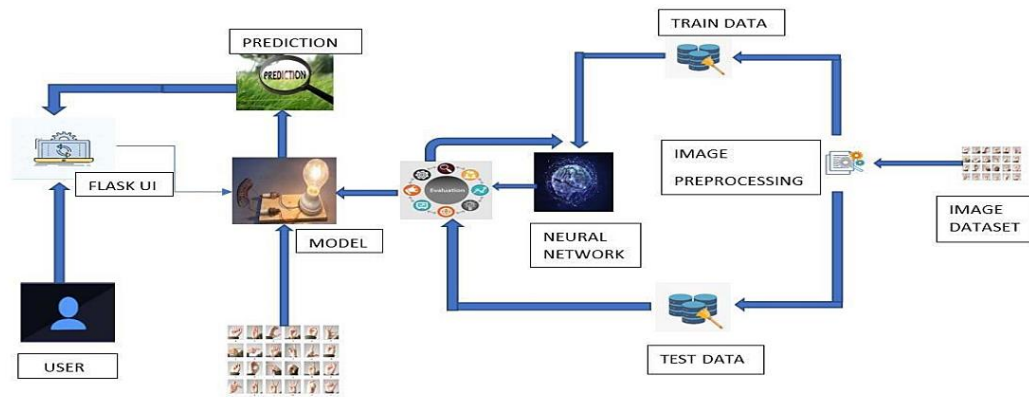
a. Data Flow Diagrams:

Dataflow Diagram:



b. Solution & Technical Architecture:

Technical Architecture:



c. User Stories:

User Stories:

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

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

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

6. PROJECT PLANNING& SCHEDULING:

a. Sprint Planning & Estimation:

Milestone Activity Plan

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

b. Sprint Delivery Schedule:

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	Collect Dataset.	9	High	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-1		USN-2	Image pre-processing	8	Medium	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-2	Model Building	USN-3	Import the required libraries, add the necessary layers and compile the model	10	High	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T

Sprint-2		USN-4	Training the image classification model using CNN	7	Medium	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-3	Training and Testing	USN-5	Training the model and testing the model's performance	9	High	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T
Sprint-4	Implementation of the application	USN-6	Converting the input sign language images into English alphabets	8	Medium	Mohanapriya C, Kavipriya G, Moulika P, Keerthana P.T

c. Report from JIRA:

The screenshot displays the Jira Software interface for a project named 'Real time communication system'. The browser address bar shows the URL: `kavipriya20.atlassian.net/jira/software/projects/RTCSPBFA/boards/1/backlog`. The interface includes a sidebar with navigation options: **PLANNING** (Roadmap, Backlog, Board) and **DEVELOPMENT** (Code, Project pages, Add shortcut, Project settings). The main content area shows the **Backlog** for the project, with a search bar and filters. A banner at the top asks: 'Does your team need more from Jira? Get a free trial of our Standard plan.' Below this, the project name is followed by a description: 'Real time communication system powered by AI for specially abled person'. The backlog is organized into sections: **RTCSPBFA Sprint 1** (0 issues) and **Backlog** (4 issues). The sprint section includes a 'Plan your sprint' card with instructions: 'Drag issues from the Backlog section, or create new issues, to plan the work for this sprint. Select **Start sprint** when you're ready.' The backlog section lists four issues, all marked as **DONE**:

- RTCSPBFA-10 Collect Dataset
- RTCSPBFA-11 Image pre-processing
- RTCSPBFA-12 Import the required libraries, add the necessary layers and compile the model
- RTCSPBFA-13 [Issue title obscured]

On the right side, there is a 'Sign in Jira Software' button and a 'Dismiss Quickstart' link. A sidebar on the far right contains a 'Show me' button and a list of links: 'Create an issue', 'Invite your teammates', 'Connect your tools', 'Get the mobile app', and 'Find help'.

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

a. Model Building:

Model Building

Import The Required Model Building Libraries

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

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

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

IMPORTING tensorflow

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

Initialize The Model

```
In [ ]: #create model
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Convolution2D
        from keras.layers import MaxPooling2D
        from keras.layers import Dropout
        from keras.layers import Flatten
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Unzipping the dataset

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

Model Building

Import The Required Model Building Libraries

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

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

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

IMPORTING tensorflow

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

Initialize The Model

```
In [ ]: #create model
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Convolution2D
        from keras.layers import MaxPooling2D
        from keras.layers import Dropout
        from keras.layers import Flatten
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Unzipping the dataset

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

Applying ImageDataGenerator to training set

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

Found 15750 images belonging to 9 classes.

Applying ImageDataGenerator to test set

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

Found 2250 images belonging to 9 classes.

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

Length of training set

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

79

Length of test set

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

12

Add Layers

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

Import The Required Model Building Libraries

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

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

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

IMPORTING tensorflow

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

Initialize The Model

```
In [ ]: #create model
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Convolution2D
        from keras.layers import MaxPooling2D
        from keras.layers import Dropout
        from keras.layers import Flatten
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Unzipping the dataset

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

Applying ImageDataGenerator to training set

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

Found 15750 images belonging to 9 classes.

Applying ImageDataGenerator to test set

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

Found 2250 images belonging to 9 classes.

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

Length of training set

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

79

Length of test set

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

12

Add Layers

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

Add The Convolution Layer

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

Model Building

Import The Required Model Building Libraries

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

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

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

IMPORTING tensorflow

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

Initialize The Model

```
In [5]: #create model
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Unzipping the dataset

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

Applying ImageDataGenerator to training set

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

Found 15750 images belonging to 9 classes.

Applying ImageDataGenerator to test set

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

Found 2250 images belonging to 9 classes.

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

Length of training set

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

79

Length of test set

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

12

Add Layers

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

Add The Convolution Layer

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

Add Pooling Layer

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

Import The Required Model Building Libraries

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

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

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

IMPORTING tensorflow

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

Initialize The Model

```
In [5]: #create model
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Convolution2D
        from keras.layers import MaxPooling2D
        from keras.layers import Dropout
        from keras.layers import Flatten
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Unzipping the dataset

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

Found 15750 images belonging to 9 classes.

Applying ImageDataGenerator to test set

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

Found 2250 images belonging to 9 classes.

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

Length of training set

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

79

Length of test set

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

12

Add Layers

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

Add The Convolution Layer

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

Add Pooling Layer

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

Add The Flatten Layer

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

Import The Required Model Building Libraries

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

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

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

IMPORTING tensorflow

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

Initialize The Model

```
In [ ]: #create model
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Convolution2D
        from keras.layers import MaxPooling2D
        from keras.layers import Dropout
        from keras.layers import Flatten
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Unzipping the dataset

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

Applying ImageDataGenerator to training set

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

Found 15750 images belonging to 9 classes.

Applying ImageDataGenerator to test set

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

Found 2250 images belonging to 9 classes.

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

Length of training set

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

79

Length of test set

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

12

Add Layers

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

Add The Convolution Layer

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

Add Pooling Layer

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

Add The Flatten Layer

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


Adding The Dense Layers

```
In [ ]: #1st hidden Layer
model.add(Dense(units=512,activation='relu'))
#2nd hidden Layer
model.add(Dense(units=256,activation='relu'))
```

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

Import The Required Model Building Libraries

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

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

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

IMPORTING tensorflow

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

Initialize The Model

```
In [ ]: #create model
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Unzipping the dataset

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

Applying ImageDataGenerator to training set

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

Found 15750 images belonging to 9 classes.

Applying ImageDataGenerator to test set

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

Found 2250 images belonging to 9 classes.

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

Length of training set

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

79

Length of test set

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

12

Add Layers

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

Add The Convolution Layer

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

Add Pooling Layer

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

Add The Flatten Layer

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

Adding The Dense Layers

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

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

Compile The Model

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

Import The Required Model Building Libraries

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

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

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

IMPORTING tensorflow

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

Initialize The Model

```
In [ ]: #create model
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Convolution2D
        from keras.layers import MaxPooling2D
        from keras.layers import Dropout
        from keras.layers import Flatten
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

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

Unzipping the dataset

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

Applying ImageDataGenerator to training set

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

Found 15750 images belonging to 9 classes.

Applying ImageDataGenerator to test set

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

Found 2250 images belonging to 9 classes.

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

Length of training set

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

79

Length of test set

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

12

Add Layers

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

Add The Convolution Layer

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

Add Pooling Layer

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

Add The Flatten Layer

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

Adding The Dense Layers

```
In [ ]: #1st hidden Layer
model.add(Dense(units=512,activation='relu'))
#2nd hidden Layer
model.add(Dense(units=256,activation='relu'))
```

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

Compile The Model

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

Fit The Mode

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

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.

Epoch 1/10
79/79 [=====] - 89s 1s/step - loss: 0.4318 - accuracy: 0.8572 - val_loss: 0.2326 - val_accuracy: 0.9471
Epoch 2/10
79/79 [=====] - 87s 1s/step - loss: 0.0413 - accuracy: 0.9886 - val_loss: 0.1702 - val_accuracy: 0.9773
Epoch 3/10
79/79 [=====] - 88s 1s/step - loss: 0.0253 - accuracy: 0.9933 - val_loss: 0.1599 - val_accuracy: 0.9764
Epoch 4/10
79/79 [=====] - 86s 1s/step - loss: 0.0086 - accuracy: 0.9979 - val_loss: 0.1979 - val_accuracy: 0.9733
Epoch 5/10
79/79 [=====] - 87s 1s/step - loss: 0.0097 - accuracy: 0.9975 - val_loss: 0.1815 - val_accuracy: 0.9782
Epoch 6/10
79/79 [=====] - 86s 1s/step - loss: 0.0067 - accuracy: 0.9982 - val_loss: 0.2445 - val_accuracy: 0.9782
Epoch 7/10
79/79 [=====] - 84s 1s/step - loss: 0.0045 - accuracy: 0.9988 - val_loss: 0.2291 - val_accuracy: 0.9782
Epoch 8/10
79/79 [=====] - 84s 1s/step - loss: 0.0083 - accuracy: 0.9973 - val_loss: 0.1956 - val_accuracy: 0.9782
Epoch 9/10
79/79 [=====] - 83s 1s/step - loss: 0.0029 - accuracy: 0.9995 - val_loss: 0.2011 - val_accuracy: 0.9773
Epoch 10/10
79/79 [=====] - 84s 1s/step - loss: 0.0027 - accuracy: 0.9991 - val_loss: 0.2367 - val_accuracy: 0.9778
```

Out[]:

Save The Model

```
In [ ]: model.save('as1png2.h5')
```

8. TESTING:

a. Test Cases:

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

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

Mounted at /content/drive

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

```
In [3]: import tensorflow as tf
        import os
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        import numpy as np
        import matplotlib.pyplot as plt
        import IPython.display as display
        from PIL import Image
        import pathlib
```

Unzipping

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

b. User Acceptance Testing:

1. Purpose of Document:

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

2. Defect Analysis:

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	11	2	3	2	18
Duplicate	1	3	4	0	8
External	3	5	0	0	8
Fixed	12	2	5	22	41
Not Reproduced	0	1	0	0	1
Skipped	0	0	1	2	3
Won't Fix	0	4	1	1	7
Totals	27	17	14	27	86

Test Case Analysis:

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	49	0	0	49
Security	4	0	0	4

Outsource Shipping	4	0	0	4
Exception Reporting	11	0	0	11
Final Report Output	2	0	0	2
Version Control	1	0	0	1

9. RESULTS:

a. Performance Metrics:

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

10.ADVANTAGES & DISADVANTAGES:

Advantages:

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

Disadvantages:

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

11.CONCLUSION:

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

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

So, to understand all sign languages, Hand gestures of deaf people by normal people this system is proposed.

12.FUTURE SCOPE:

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

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

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

13. APPENDIX:

Source Code:

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



```

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

```

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta name="viewport" content="width=device-width, initial-scale=1">
5  <style>
6  body {font-family: Arial, Helvetica, sans-serif;}
7
8  /* Full-width input fields */
9  input[type=text], input[type=password] {
10     width: 100%;
11     padding: 12px 20px;
12     margin: 8px 0;
13     display: inline-block;
14     border: 1px solid #ccc;
15     box-sizing: border-box;
16 }
17
18 /* Set a style for all buttons */
19 button {
20     background-color: #273298;
21     color: white;
22     padding: 14px 20px;
23     margin: 8px 0;
24     border: none;
25     cursor: pointer;
26     width: 100%;
27 }
28
29 button:hover {
30     opacity: 0.8;
31 }

```

```

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 {

```

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta name="viewport" content="width=device-width, initial-scale=1">
5  <style>
6  body {font-family: Arial, Helvetica, sans-serif;}
7
8  /* Full-width input fields */
9  input[type=text], input[type=password] {
10     width: 100%;
11     padding: 12px 20px;
12     margin: 8px 0;
13     display: inline-block;
14     border: 1px solid #ccc;
15     box-sizing: border-box;
16 }
17
18 /* Set a style for all buttons */
19 button {
20     background-color: #273298;
21     color: white;
22     padding: 14px 20px;
23     margin: 8px 0;
24     border: none;
25     cursor: pointer;
26     width: 100%;
27 }
28
29 button:hover {
30     opacity: 0.8;
31 }
32
33 .cancelbtn {
34     width: auto;
35     padding: 10px 18px;
36     background-color: #f44336;
37 }
38
39 /* Center the image and position the close button */
40 .imgcontainer {
41     text-align: center;
42     margin: 24px 0 12px 0;
43     position: relative;
44 }
45
46 .img.avatar {
47     width: 40%;
48     border-radius: 50%;
49 }
50
51 .container {
52     padding: 16px;
53 }
54
55 span.psw {
56     float: right;
57     padding-top: 16px;
58 }
59
60 /* The Modal (background) */
61 .modal {
62     display: none; /* Hidden by default */
63     position: fixed; /* Stay in place */
64     z-index: 1; /* Sit on top */
65     left: 0;
66     top: 0;
67     width: 100%; /* Full width */
68     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 }
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 }

```

```

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

```

```

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 style="text-align: center; color: rgba(1, 2, 2, 0.874);">REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED</h2>
131
132 <button onclick="document.getElementById('id01').style.display='block'">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       
140     </div>

```

```

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>

```



```

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 {
250     position: absolute;
251     right: 20px;
252     top: 20px;

```

```

253     display: flex;
254 }
255
256 .controls > button {
257     width: 45px;
258     height: 45px;
259     text-align: center;
260     border-radius: 100%;
261     margin: 0 6px;
262     background: transparent;
263 }
264
265 .controls > button:hover svg {
266     color: #white !important;
267 }
268
269 @media (min-width: 300px) and (max-width: 400px) {
270     .controls {
271         flex-direction: column;
272     }
273
274     .controls button {
275         margin: 5px 0 !important;
276     }
277 }
278
279 .controls > button > svg {
280     height: 20px;
281     width: 18px;
282     text-align: center;
283     margin: 0 auto;
284     padding: 0;
285 }
286
287 .controls button:nth-child(1) {
288     border: 2px solid #1a12b3;
289 }

```

```

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();
336
337 const controls = document.querySelector('.controls');
338 const cameraOptions = document.querySelector('.video-options>select');
339 const video = document.querySelector('video');
340 const canvas = document.querySelector('canvas');
341 const screenshotImage = document.querySelector('img');
342 const buttons = [...controls.querySelectorAll('button')];
343 let streamStarted = false;
344
345 const [play, pause, screenshot] = buttons;
346
347 const constraints = {
348     video: {
349         width: {
350             min: 1280,
351             ideal: 1920,
352             max: 2560,
353         },
354         height: {
355             min: 720,
356             ideal: 1080,
357             max: 1440
358         },
359     }
360 };
361 </script>
362 <script>
363 const getCameraSelection = async () => {
364     const devices = await navigator.mediaDevices.enumerateDevices();

```



```

365     const videoDevices = devices.filter(device => device.kind === 'videoinput');
366     const options = videoDevices.map(videoDevice => {
367       return `<option value="${videoDevice.deviceId}">${videoDevice.label}</option>`;
368     });
369     cameraOptions.innerHTML = options.join('');
370   };
371
372   </script>
373   <script>
374
375   play.onclick = () => {
376     if (streamStarted) {
377       video.play();
378       play.classList.add('d-none');
379       pause.classList.remove('d-none');
380       return;
381     }
382     if ('mediaDevices' in navigator && navigator.mediaDevices.getUserMedia) {
383       const updatedConstraints = {
384         ...constraints,
385         deviceId: {
386           exact: cameraOptions.value
387         }
388       };
389       startStream(updatedConstraints);
390     }
391   };
392
393   const startStream = async (constraints) => {
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>
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
7     <title>SmartBridge WebApp.VideoTemplate</title>
8     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
9     <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
10    <link rel="stylesheet" href="assets/css/Banner-Heading-Image.css">
11    <link rel="stylesheet" href="assets/css/Navbar-Centered-Brand.css">
12    <link rel="stylesheet" href="assets/css/styles.css">
13 </head>
14
15 <body style="background: #394348;">
16     <nav class="navbar navbar-light navbar-expand-md py-3" style="background: #212529;">
17         <div class="container">
18             <div></div><a class="navbar-brand d-flex align-items-center" href="#"><span
19                 class="bs-icon-sm bs-icon-rounded bs-icon-primary d-flex justify-content-center align-items-center me-2 bs-icon"><i
20                     class="fas fa-flask"></i></span><span style="color: #212529;">Real-Time Communication
21                 System Powered By AI&nbsp;For Specially Abled</span></a>
22             <div></div>
23         </div>
24     </nav>
25     <section>
26         <div class="d-flex flex-column justify-content-center align-items-center">
27             <div class="d-flex flex-column justify-content-center align-items-center" id="div-video-feed"
28                 style="width: 640px; height: 480px; margin: 10px; min-height: 480px; min-width: 640px; border-radius: 10px; border: 4px dashed #212529;">
29                 
31             </div>
32         </div>
33         <div class="d-flex flex-column justify-content-center align-items-center" style="margin-bottom: 10px;"><button
34             class="btn btn-info type="button" data-bs-target="#modal-1" data-bs-toggle="modal">Quick Reference
35             <strong>ASL Alphabets</strong></button></div>
36     </section>
37 </body>

```

```

38     <div class="container">|
39         <div class="accordion text-white" role="tablist" id="accordion-1">
40             <div class="accordion-item" style="background: □rgb(33,37,41);">
41                 <h2 class="accordion-header" role="tab"><button class="accordion-button" data-bs-toggle="collapse"
42                     data-bs-target="#accordion-1 .item-1" aria-expanded="true"
43                     aria-controls="accordion-1 .item-1"
44                     style="background: □rgb(39,43,48);color: ■rgb(255,255,255);">About The Project</button></h2>
45                 <div class="accordion-collapse collapse show item-1" role="tabpanel" data-bs-parent="#accordion-1">
46                     <div class="accordion-body">
47                         <p class="mb-0">Artificial Intelligence has made it possible to handle our daily activities
48                             in new and simpler ways. With the ability to automate tasks that normally require human
49                             intelligence, such as speech and voice recognition, visual perception, predictive text
50                             functionality, decision-making, and a variety of other tasks, AI can assist people with
51                             disabilities by significantly improving their ability to get around and participate in
52                             daily activities.<br><br>Currently, Sign Recognition is available <strong>only for
53                             alphabets A-I</strong> and not for J-Z, since J-Z alphabets also require Gesture
54                             Recognition for them to be able to be predicted correctly to a certain degree of
55                             accuracy.</p>
56                     </div>
57                 </div>
58             </div>
59             <div class="accordion-item" style="background: □rgb(33,37,41);">
60                 <h2 class="accordion-header" role="tab"><button class="accordion-button collapsed"
61                     data-bs-toggle="collapse" data-bs-target="#accordion-1 .item-2" aria-expanded="false"
62                     aria-controls="accordion-1 .item-2"
63                     style="background: □rgb(39,43,48);color: ■rgb(231,241,255);">Developed By</button></h2>
64                 <div class="accordion-collapse collapse item-2" role="tabpanel" data-bs-parent="#accordion-1">
65                     <div class="accordion-body">
66                         <p class="mb-0">Students at VIT-Bhopal University during SmartBridge AI Externship
67                             Program.<br><br>1. <strong>Nirlov Deb</strong> 19BCG10067<br>2.
68                             <strong>Kushagra</strong> 19BCG10025<br>3. <strong>Kartik Dhasmana</strong> 19BCG10002
69                         </p>
70                     </div>
71                 </div>
72             </div>
73         </div>
74     </div>
75 </section>
76 <div class="modal fade" role="dialog" tabindex="-1" id="modal-1">
77     <div class="modal-dialog" role="document">
78         <div class="modal-content">
79             <div class="modal-header">
80                 <h4 class="modal-title">American Sign Language - Alphabets</h4><button type="button"
81                     class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
82             </div>
83             <div class="modal-body"></div>
84             <div class="modal-footer"><button class="btn btn-secondary" type="button"
85                 data-bs-dismiss="modal">Close</button></div>
86         </div>
87     </div>
88 </div>
89 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
90 </body>
91
92 </html>

```

```
1  .bs-icon {
2    --bs-icon-size: .75rem;
3    display: flex;
4    flex-shrink: 0;
5    justify-content: center;
6    align-items: center;
7    font-size: var(--bs-icon-size);
8    width: calc(var(--bs-icon-size) * 2);
9    height: calc(var(--bs-icon-size) * 2);
10   color: var(--bs-primary);
11 }
12
13 .bs-icon-xs {
14   --bs-icon-size: 1rem;
15   width: calc(var(--bs-icon-size) * 1.5);
16   height: calc(var(--bs-icon-size) * 1.5);
17 }
18
19 .bs-icon-sm {
20   --bs-icon-size: 1rem;
21 }
22
23 .bs-icon-md {
24   --bs-icon-size: 1.5rem;
25 }
26
27 .bs-icon-lg {
28   --bs-icon-size: 2rem;
29 }
30
31 .bs-icon-xl {
32   --bs-icon-size: 2.5rem;
33 }
34
35 .bs-icon.bs-icon-primary {
36   color: var(--bs-white);
37   background: var(--bs-primary);
```



```

38 }
39
40 .bs-icon.bs-icon-primary-light {
41   color: var(--bs-primary);
42   background: rgba(var(--bs-primary-rgb), .2);
43 }
44
45 .bs-icon.bs-icon-semi-white {
46   color: var(--bs-primary);
47   background: rgba(255, 255, 255, .5);
48 }
49
50 .bs-icon.bs-icon-rounded {
51   border-radius: .5rem;
52 }
53
54 .bs-icon.bs-icon-circle {
55   border-radius: 50%;
56 }

```

REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED

Login



Username

Enter Username

Password

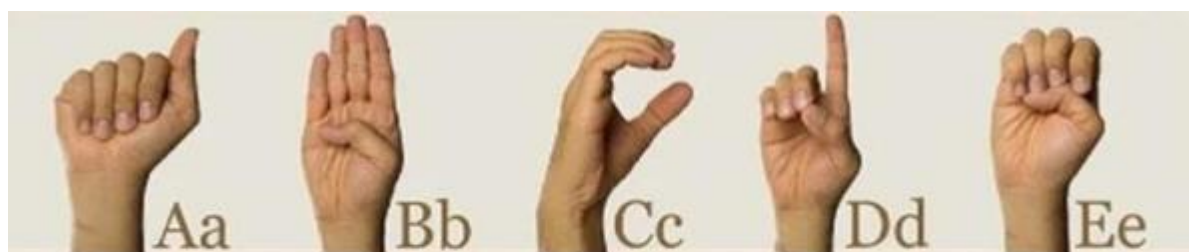
Enter Password

Login

☒ Remember me

Cancel

[Forgot password?](#)



GITHUB LINK:

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