Project Report

Team ID	PNT2022TMID09247
Project Name	Real-Time Communication System Powered by AI for Specially Abled
Team Lead	Chadalavada Gautham (Roll No. 310619104018)
Team Member 1	Lakshman KR (Roll No. 310619104059)
Team Member 2	Badrinath K (Roll No. 310619104015)
Team Member 3	CB Sathvik (Roll No. 310619104017)

1. INTRODUCTION

1.1. PROJECT OVERVIEW

The project developed is a system that converts hand gestures of a Deaf-Mute individual into its respective ASL (American Sign Language) alphabets for a normal individual for communication. The main customer for our project are: People who want to communicate with deaf-mute individual who desire to communicate with others, and deaf-mute individual who desire to communicate with others. This project tries to solve the communication during the time of emergencies. The project is developed on Python Platform using CNN (Convolutional Neural Network) model from TensorFlow package.

1.2. PURPOSE

Everybody cannot afford to have a human translators of sign language, they may not be available all the time and they are quite expensive. People who engage in conversation with deaf-mute individual will find it hard and tedious. Deaf-mute individual may lose a lot of opportunities because they cannot speak or express their thoughts verbally in situations like an interview. This project aims to overcome the said challenges.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEMS

- Existing system (or) frameworks has too many false positives. The system predicts the gestures inaccurately.
- Real Time recognition of gestures into text/speech and text/speech into gestures is not available.

2.2. REFERENCES

[1] Saed Mian Qaisar, Sarah Niyazi, Abdulhamit Subasi, “Efficient Isolated Speech to Sign Conversion Based on the Adaptive Rate Processing”; Procedia Computer Science, Vol. 163, PP. 35–40, 2019.

[2] T. Bohra, S. Sompura, K. Parekh and P. Raut, "Real-Time Two Way Communication System for Speech and Hearing Impaired Using Computer Vision and Deep Learning" International Conference on Smart Systems and Inventive Technology (ICSSIT), pp. 734-739, 2019.

[3] Ma, Jiyong, Wen Gao, Jiangqin Wu, and Chunli Wang. "A continuous Chinese Sign Language recognition system." In Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580), pp. 428-433. IEEE, 2000.

[4] Vogler, C., and D. Handshapes Metaxas. "Movements: Multiple-Channel American Sign Language Recognition." Gesture-Based Communication in Human-Computer Interaction. Lecture Notes in Computer Science: 247-258.

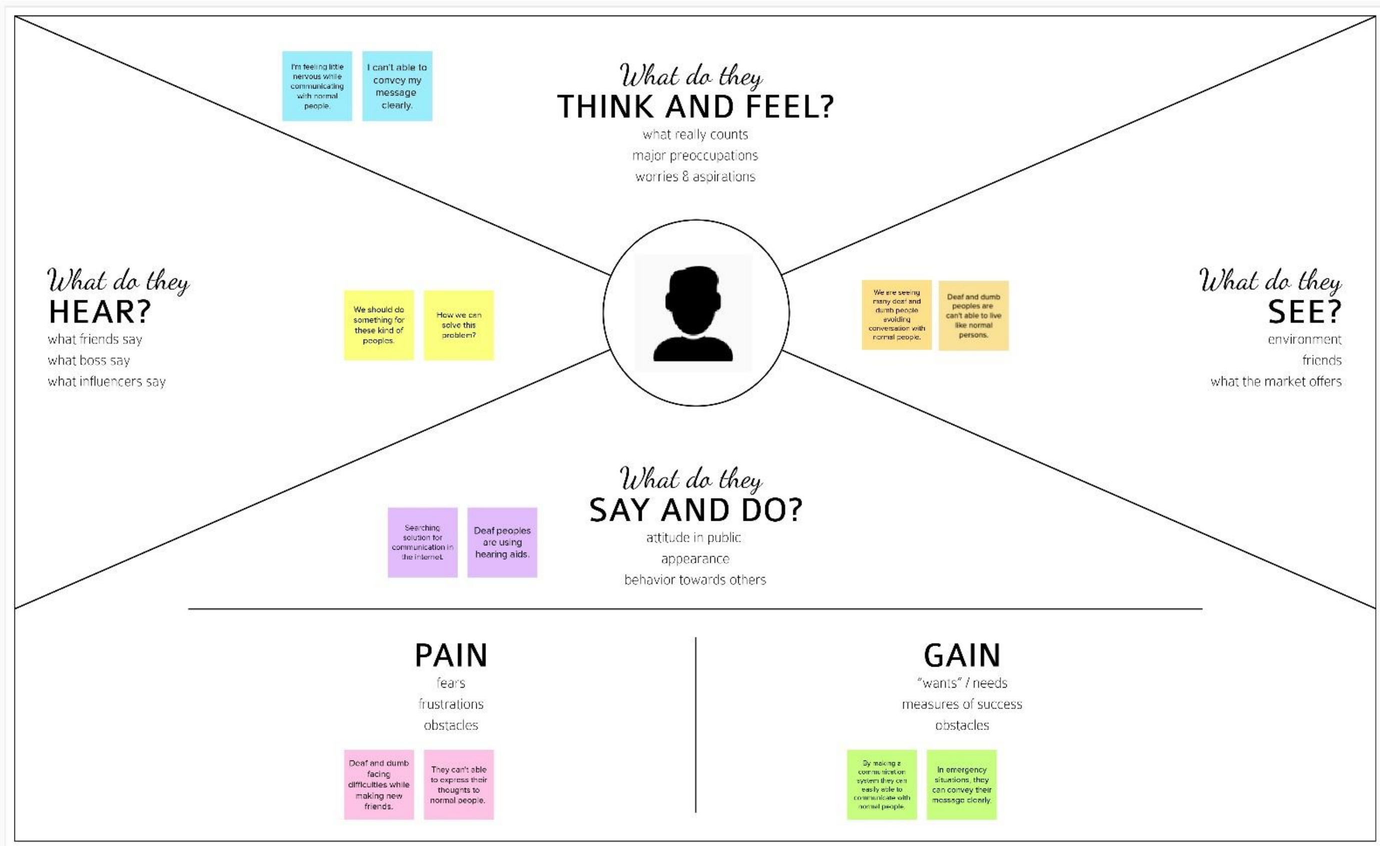
[5] Pavlovic, V, Sharma, R., &Huang T., "Visual Interpretation of Hand Gestures for Human-Computer Interaction (HCI): A Review", IEEE TOPAMI, VOL. 19, NO. 7, 1999.

2.3. PROBLEM STATEMENT DEFINITION

The study of human-computer interaction has shown a great deal of interest in hand gesture recognition. In many areas of human-computer interaction, including virtual reality, gaming, automobile system control, and robotic control, quick and precise hand gesture recognition is crucial. As more sensors are added, there are numerous different ways to categorise hand motions. Since gesture identification is a problem of image classification and 2D CNNs are effective in image classification, we have chosen to employ a convolutional neural network for this task. A system that converts the sign language into the respective ASL (American Sign Language) alphabet to convey a message to normal people is developed in this project.

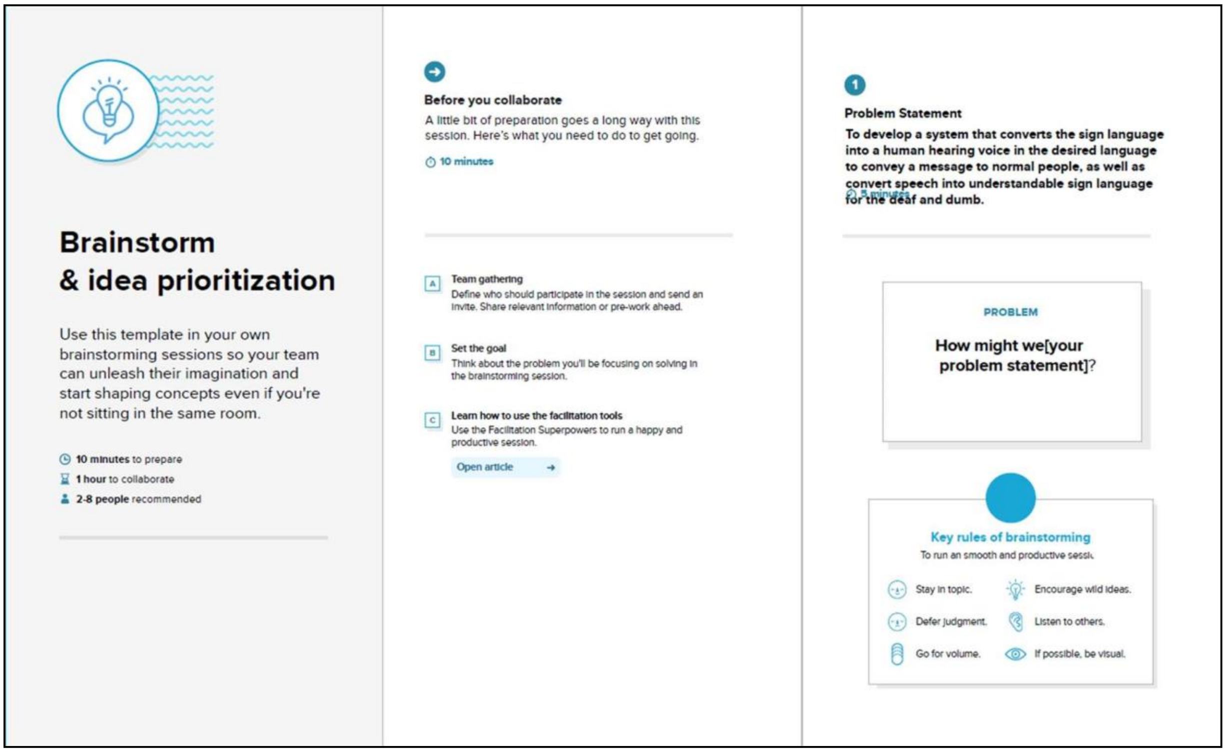
3. IDEATION & PROPOSED SOLUTION

3.1. EMPATHY MAP CANVAS

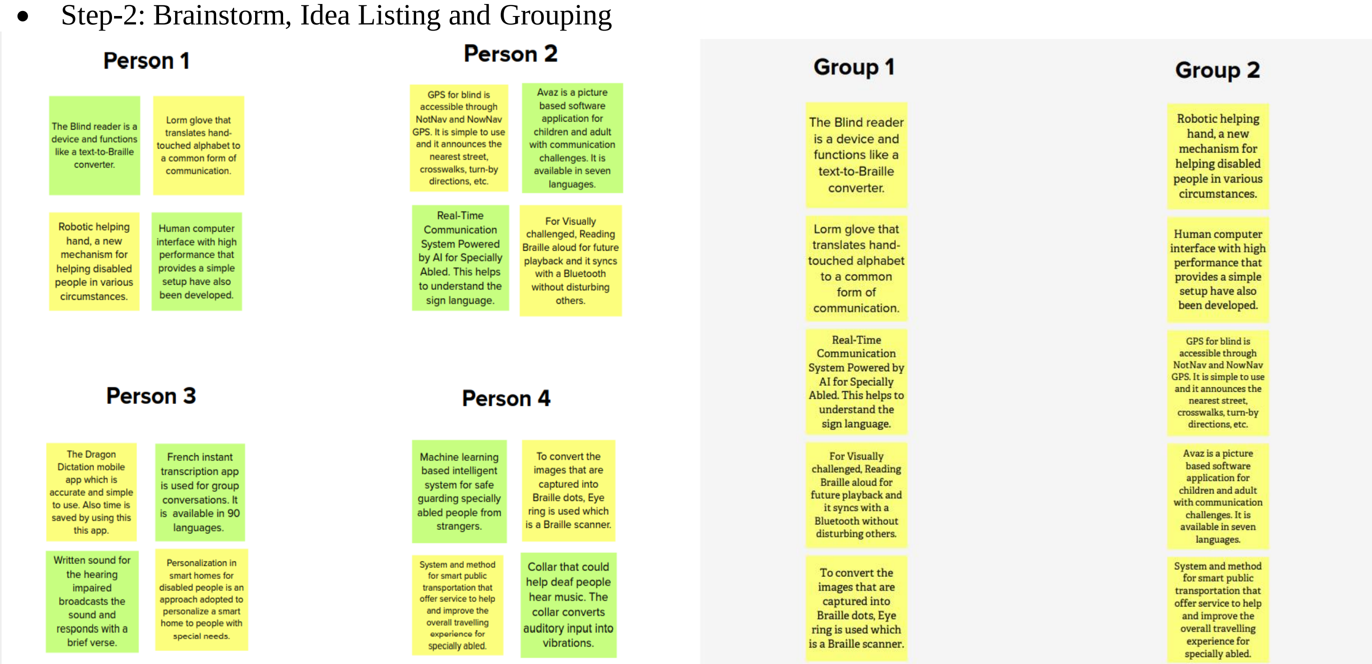


3.2. IDEATION & BRAINSTORMING

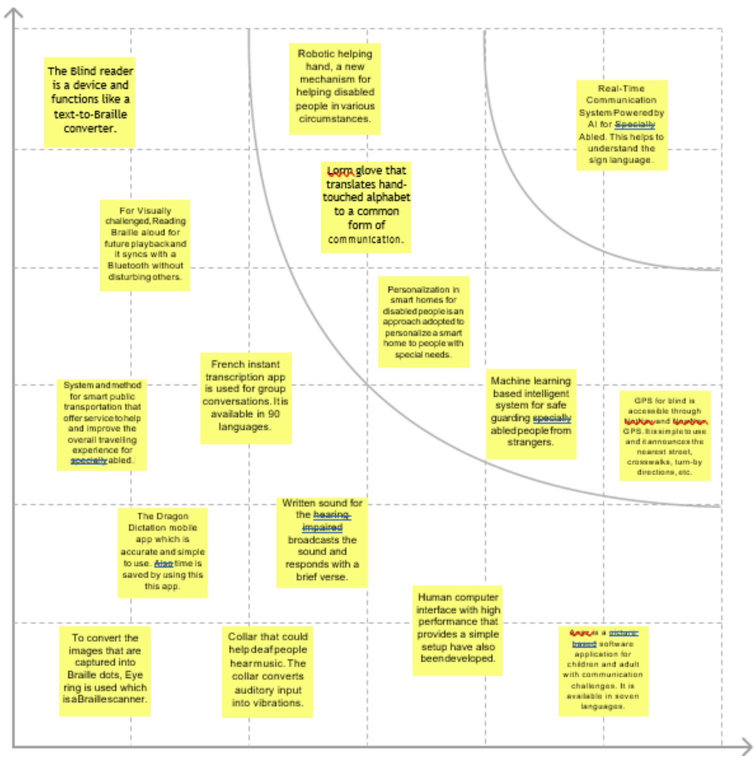
- Step-1: Team Gathering, Collaboration and Select the Problem Statement



- Step-2: Brainstorm, Idea Listing and Grouping



- Step-3: Idea Prioritization



3.3. PROPOSED SOLUTION

S No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Matching people with speech and hearing problems with normal people has always been a difficult task. People with language disabilities have difficulty getting their message across to normal people. Because normal people don't understand sign language. Communication in times of crisis is a pain. Not everyone can tolerate artificial sign language interpreters. People with speech and hearing problems may miss out on many opportunities because they cannot speak or speak orally in situations such as meetings.
2.	Idea / Solution description	1. Matching people with speech and hearing problems with normal people has always been a difficult task. People with language disabilities have difficulty getting their message across to normal people. Because normal people don't understand sign language. 2. Design and implement a system using artificial intelligence, convolutional neural networks, computer vision and image processing, taking gestures as inputs and converting them into speech/text.

3.	Novelty / Uniqueness	1. An app called “Mozhi” that uses computer vision, Artificial Intelligence convolutional neural network and image processing. 2. It recognizes the image of sign language from the speaker and then translates it into speech/text.
4.	Social Impact / Customer Satisfaction	1. Disabled people experience a great deal of difficulty with day-to-day activities. 2. The primary aim of this application is to make speech-impaired individual work independently.
5.	Business Model (Revenue Model)	1. AI can create income through direct customers and collaborate with health care sector and produce income from their customers. 2. Speech-hearing impaired employees of B2B services can use the app to pass messages concurring on to the organization.
6.	Scalability of the Solution	1. AI innovation assists the specially abled with opening up new doors for availability consideration in the public eye and independent living. 2. 2. It might also open the door to more cutting-edge and creative innovations to the most challenging problems faced by the specially abled.

3.4. PROBLEM SOLUTION FIT

4. REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENT

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 Upload image through gallery
FR-4	Text to speech	Select speech icon to convert the respective text for sign language
FR-5	Whiteboard	Use whiteboard to share the message by drawing

FR-6	Emergency templates	Select emergency templates icon to pass the message quickly

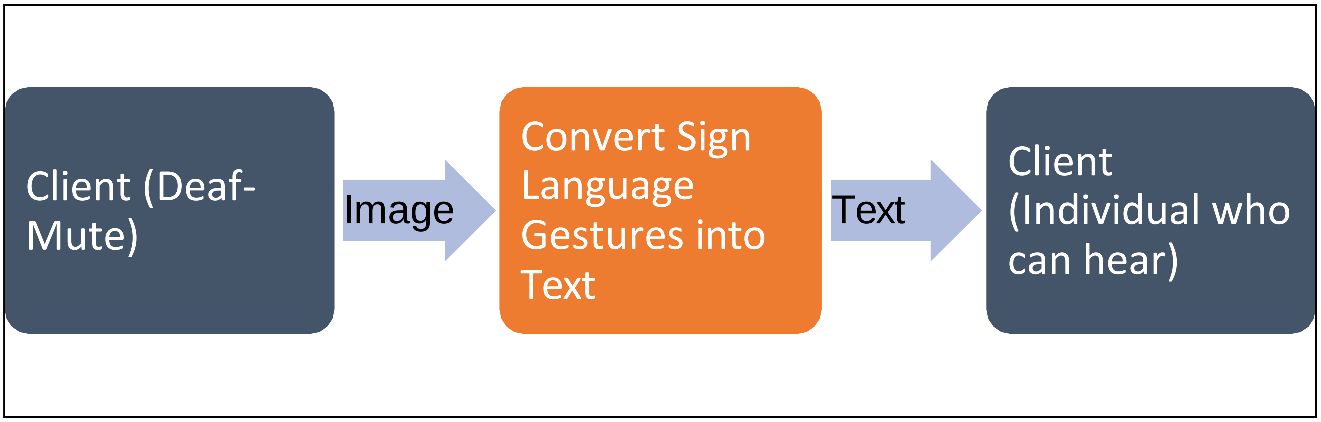
4.2. NON-FUNCTIONAL REQUIREMENT

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Client can undoubtedly upload the image and this application is planned in a manner here, client can without much of a stretch discover some predefined layouts
NFR-2	Security	Client should sign in into an app only then proceed for further process. So unapproved access will be kept away from at max.
NFR-3	Reliability	This application has robust adaptation to non-critical failure and regardless of whether an error happens likewise it recuperates rapidly.
NFR-4	Performance	This application will rapidly transfer and process the images since it predicts the gestures through signing utilizing CNN model and it gives high accuracy.
NFR-5	Availability	The predefined formats will be accessible to all clients and furthermore have whiteboard choice. This application is planned such that it is straightforward and accessible to all clients.
NFR-6	Scalability	Engineers can add new formats and it will build adaptability and this application has premium elements where client approach google maps and google duo.

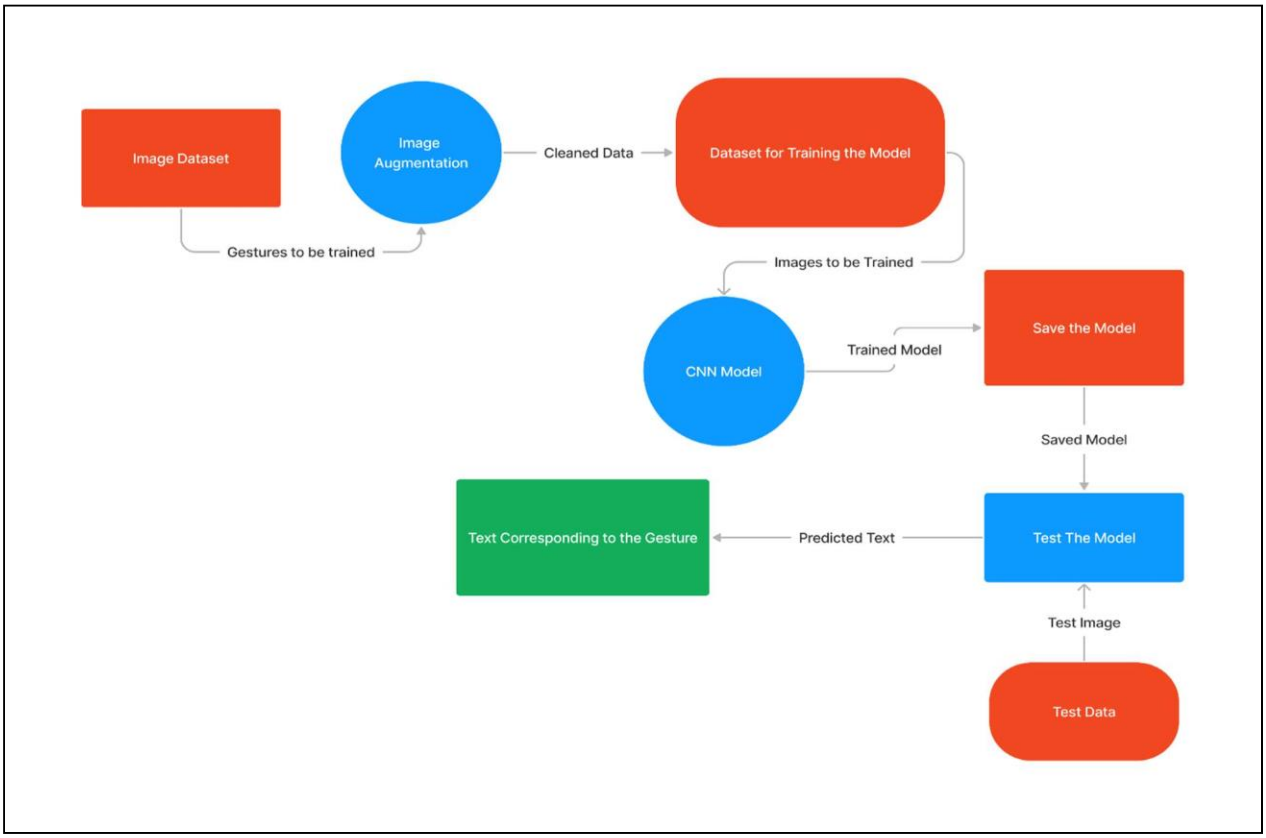
5. PROJECT DESIGN

5.1. DATA FLOW DIAGRAMS

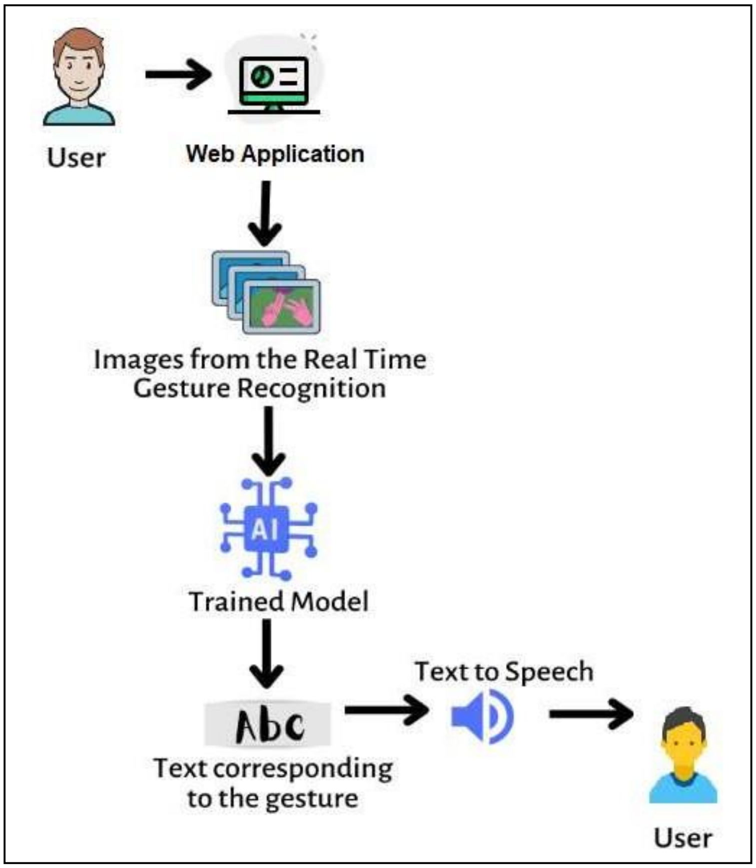
- Level 0 - DFD



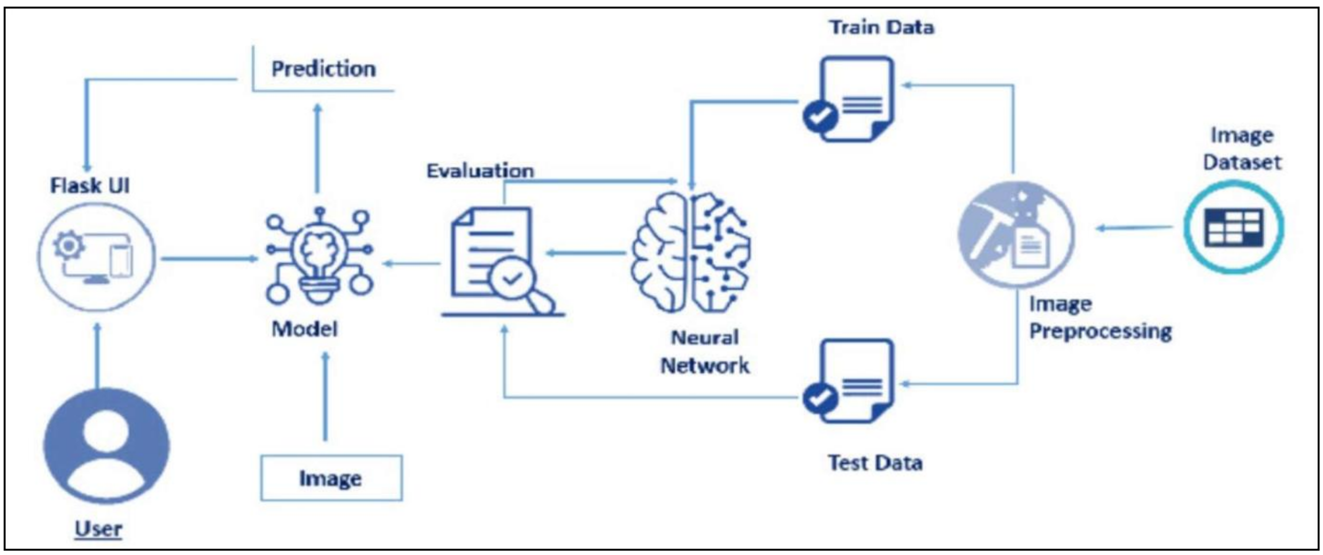
- Level 1 - DFD



- Simplified Flow Diagram



5.2. SOLUTION & TECHNICAL ARCHITECTURE



5.3. USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria
Customer (People who cannot hear)	Convert sign language into text	USN - 1	As a user, I can open camera in the app and record my signs to be converted into text	I can communicate with normal people effectively

		USN - 2	As a user, I can upload my previous sign gestures to communicate faster	I can have a list of frequently used signs to make for fast reference
	Dashboard	USN – 3	Buttons to record the signs, to convert in real-time and other buttons should be available in the right places	All features must be easily accessible
		USN – 4	Emergency calls must be available so that I can press a button in times of emergency to get the attention of others	I can feel safe because of the Emergency Button which can get me help.
Customer (People who can hear and talk)	Convert sign language into text	USN - 5	As a user, I can open back camera in the app and record the specially abled people's signs to be converted into text	I can understand the mode of communication of specially-abled people effectively
	Show the message to convey in the form of text	USN - 6	As a user, I can open a Text-pad that is available in the app, so that the deaf people can see the message I need to convey	I can convey my message to them effectively
Administrator	Integrate application with trained model	USN - 7	As an admin, I should be able to integrate the AI model into the application and maintain the application	I can give best experience to app users

6. PROJECT PLANNING & SCHEDULING

6.1. SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority

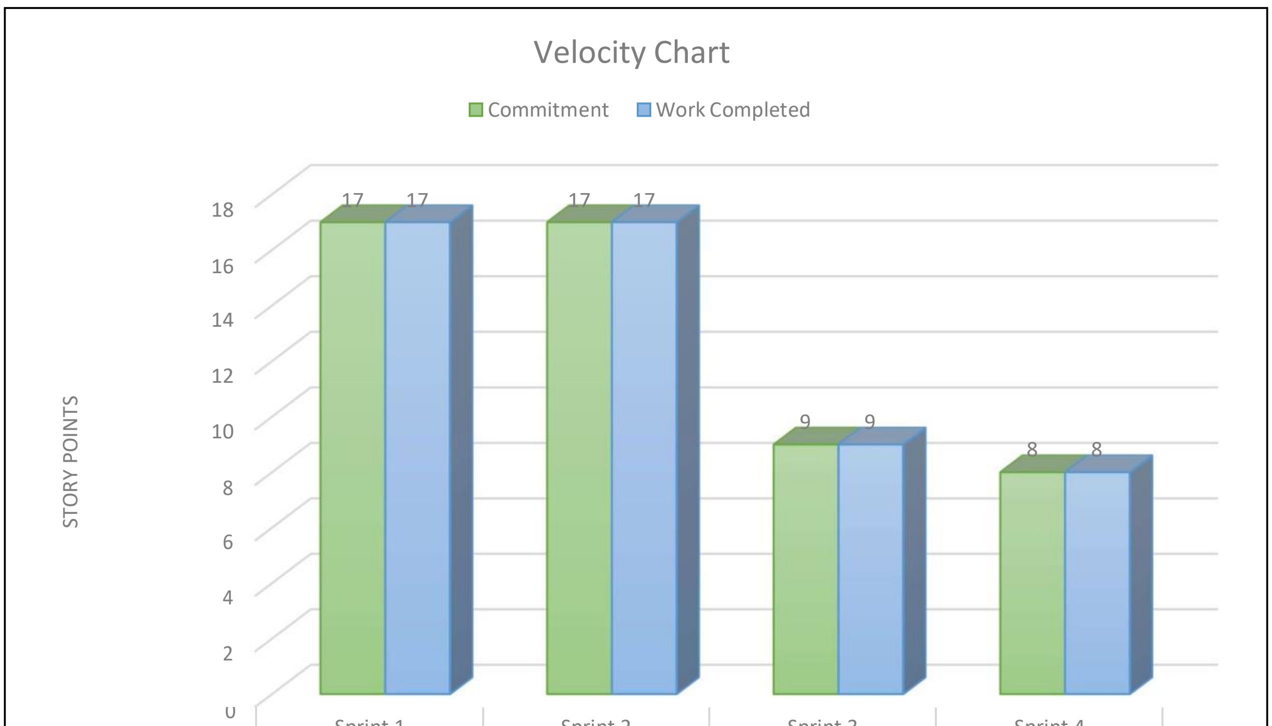
Sprint-1	Dataset Collection	USN-1	Collect Dataset for building model.	9	High
Sprint-1	Image Pre-processing	USN-2	Perform Pre-processing techniques on the dataset.	8	Medium
Sprint-2	Model Building	USN-3	Import the required libraries, add the necessary layers and compile the model.	10	High
Sprint-2		USN-4	Training the image classification model using CNN.	7	Medium
Sprint-3	Training and Testing the Model	USN-5	Training the model and testing the model's performance.	9	High
Sprint-4	Application Development	USN-6	Converting the input gesture image into English Alphabets.	8	Medium

6.2. SPRINT DELIVERY 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	17	6 Days	24 Oct 2022	29 Oct 2022	17	29 Oct 2022
Sprint-2	17	6 Days	31 Oct 2022	05 Nov 2022	17	05 Nov 2022
Sprint-3	9	6 Days	07 Nov 2022	12 Nov 2022	9	12 Nov 2022
Sprint-4	8	6 Days	14 Nov 2022	19 Nov 2022	8	19 Nov 2022

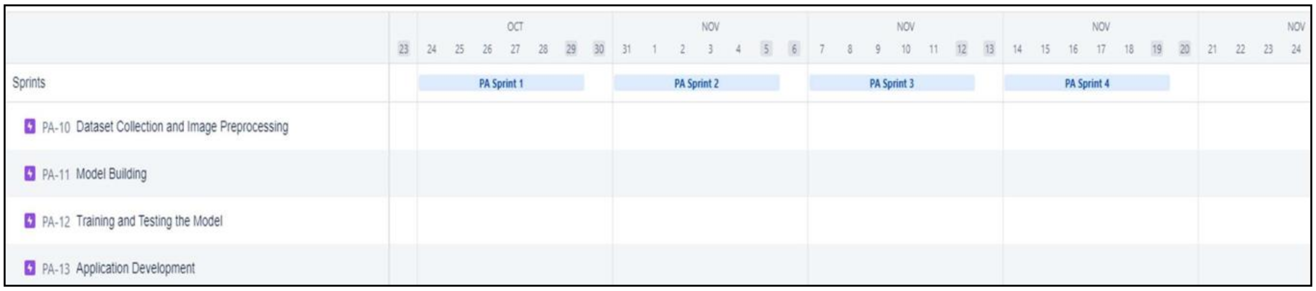
- Burndown Chart

- Velocity Chart



	Sprint 1	Sprint 2	Sprint 3	Sprint 4
Commitment	17	17	9	8
Work Completed	17	17	9	8

6.3. REPORTS FROM JIRA



7. CODING & SOLUTIONING

7.1. IMAGE PREPROCESSING

- Image pre-processing includes zooming, shearing, flipping to increase the robustness of the model after it is built. Keras package is used for pre-processing images.
- Importing ImageDataGenerator Library to create an instance for which include shearing, rescale, zooming, etc to make the model robust with different types of images.

```
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,vertical_flip=False)

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

- Applying ImageDataGenerator Functionality To Train And Test Set

```
In [2]: x_train = train_datagen.flow_from_directory('dataset/training_set',target_size=(64,64),batch_size=300,class_mode='categorical',co

        x_test = test_datagen.flow_from_directory('dataset/test_set',target_size=(64,64),batch_size=300,class_mode='categorical',color_mo

Found 15750 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

7.2. MODEL BUILDING

Import The Required Model Building Libraries

```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

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

```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

```
import tensorflow as tf
import os
```

Initialize The Model

```
#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
```

```
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

```
!unzip '/content/conversation engine for deaf and dumb (1).zip'
```

Applying ImageDataGenerator to training set

```
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

```
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.
```

```
a=len(x_train)
b=len(x_test)
```

Length of training set

```
print(a)
```

```
79
```

Length of test set

```
print(b)
```

```
12
```

Add Layers

```
#create model
model=Sequential()
```

Add The Convolution Layer

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

Add Pooling Layer

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

7.3.TESTING THE MODEL

- Importing The Packages and Loading the Saved Model

```
#import imagedatagenerator
from keras.preprocessing.image import ImageDataGenerator
```

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

```
#testing datagen
test_datagen=ImageDataGenerator(rescale=1./255)
```

IMPORTING tensorflow

```
import tensorflow as tf
import os
```

- Loading the Test Image, Pre-Processing it And Prediction

```
In [19]: 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)
             predictions = model.predict_classes(img)
             print(predictions)

In [21]: frame = cv2.imread(r"dataset/test_set/G/1.png")
         data = detect(frame)

         [[1.1529493e-09 1.6801257e-12 3.0758306e-07 3.6168924e-08 2.1814937e-11
           6.9361130e-09 9.9995184e-01 4.7746969e-05 3.6307211e-09]]
         [6]
```

- The output [6] in the above image represents the index value in the array [‘A’,’B’,’C’,’D’,’E’,’F’,’G’,’H’,’I’].
- Thus, the predicted alphabet is G.

7.4.FLASK APPLICATION

- Loading the required packages

```
import numpy as np
import cv2
import os
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.backend import set_session
from flask import Flask, render_template,Response
import tensorflow as tf
from gtts import gTTS
global graph
global writer
from skimage.transform import resize
```

- Initializing graph, loading the model, initializing the flask app and loading the video.
- Graph element is required to work with TensorFlow. So, graph element is created explicitly.

```
graph = tf.get_default_graph()
model = load_model('signlanguage1.h5')
vals = ['A','B','C','D','E','F','G','H','I']
app = Flask(__name__)
print("[INFO] accessing video stream...")
camera = cv2.VideoCapture(1)
camera.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
pred=""
```

- Configuring the home page

```
@app.route('/')
def index():
    return render_template('index.html')
```

- Pre-processing the frames captured from the camera

```
def detect(frame):
    global pred
    global graph
    img = resize(frame,(64,64,1))
    x = image.img_to_array(img)
    x = np.expand_dims(x,axis=0)
    with graph.as_default():
        predictions = model.predict_classes(x)
    print(predictions)
    pred=vals[predictions[0]]
    print(pred)
```

- Video Feed call from the HTML PAGE

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

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

7.5.HTML PAGE

- HTML page to display the processed video on the screen, so that the person can show signs which can be detected.

```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>Hearsome</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="static/css/Banner-Heading-Image.css">
    <link rel="stylesheet" href="static/css/Navbar-Centered-Brand.css">
    <link rel="stylesheet" href="static/css/styles.css">
</head>
<body>

<nav class="navbar navbar-light navbar-expand-md py-3" style="background: #370080;">
        <div class="container">
            <div></div><a class="navbar-brand d-flex align-items-center" href="#"><h4 style="color: #ffffff; font-style: oblique; text-align: center;font-family: Arial Black"><strong> Real-Time Communication
                    System Powered By AI For Specially Abled - Team ID: PNT2022TMID09247</strong></h4></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: 800px;height: 600px;margin: 10px;min-height: 480px;min-width: 640px;border-radius: 10px;border: 5px groove #000000 ;">
                <img src="{{ url_for('video_feed') }}" style="width: 100%;height: 100%;color: rgb(255,255,255);text-align: center;font-size: 20px;"
                    alt="Camera Access Not Provided!">
            </div>
        </div>
    </section>
<section>
        <div class="container">
            <div class="accordion text-white" role="tablist" id="accordion-1">

                <div class="accordion-item" style="font-style: oblique; background: #a796b8;">
                    <h2 class="accordion-header" role="tab"><button class="accordion-button collapsed"
                            data-bs-toggle="collapse" data-bs-target="#accordion-1 .item-2" aria-expanded="false"
                            aria-controls="accordion-1 .item-2"
                            style="font-style: oblique; background: #4d0074;color: #ffffff;">Project: Real-Time Communication System Powered by AI for Specially Abled</button></h2>
                    <div class="accordion-collapse collapse item-2" role="tabpanel" data-bs-parent="#accordion-1">
                        <div class="accordion-body">
                            <p class="mb-0">Team ID: PNT2022TMID09247<br><br>Hearsome Developed By,<br><br>1. <strong>Chadalavada Gautham</strong> 310619104018<br>2.
                                <strong>Badrinath K</strong> 310619104015<br>3. <strong>Lakshman K.R.</strong> 310619104059<br>4. <strong> Sathvik Canchibalaji</strong> 310619104017
                            </p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </section>
    <div class="modal fade" role="dialog" tabindex="-1" id="modal-1">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">American Sign Language - Alphabets</h4><button type="button"
                        class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
                </div>
                <div class="modal-body"><img src="{{ url_for('static', filename='img/ASL_Alphabets.png') }}" width="100%"></div>
                <div class="modal-footer"><button class="btn btn-secondary" type="button"
                        data-bs-dismiss="modal">Close</button></div>
            </div>
        </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

8. TESTING

8.2.TEST CASES

Test Case ID	Test Scenario	Steps to Execute	Expected Result	Actual Result
1	Verify if user is able to provide camera access.	1. Enter URL and click go. 2. Give Camera Access.	Camera is On.	Working as expected.
2	Verify if user is able to get the desirable prediction for the gesture.	1. Enter URL and click go. 2. Give Camera Access. 3. Make Gesture in front of camera.	Alphabet is predicted for the gesture.	Working as expected.

8.3.USER ACCEPTANCE TESTING

- Defect Analysis

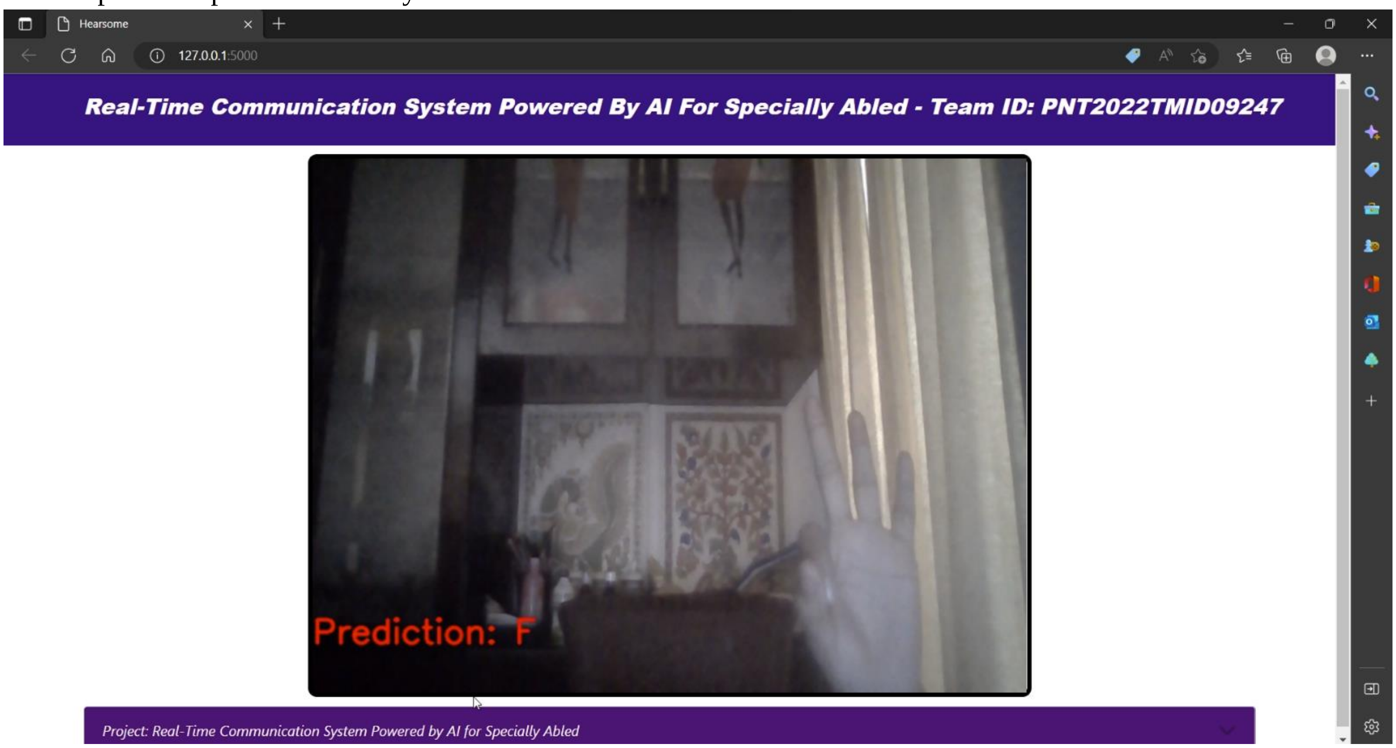
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	12	1	1	14
External	5	0	0	0	5
Fixed	11	3	2	2	18
Skipped	0	0	2	0	2

Won’t Fix	4	0	0	0	4
Totals	20	15	5	3	43

- Test Case Analysis

Section	**Total Cases**	**Not Tested**	**Fail**	**Pass**
Client Application	5	1	0	5
Security	2	0	0	2
Exception Reporting	2	0	0	2
Final Report Output	9	0	0	9

- The project developed was tested by an end user and the application converts the gestures to its respective alphabet accurately.



9. RESULTS

9.2.PERFORMANCE METRICS

- Model Summary

```
In [40]: 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
_________________________________________________________________
```

- Confusion Matrix and Classification Report

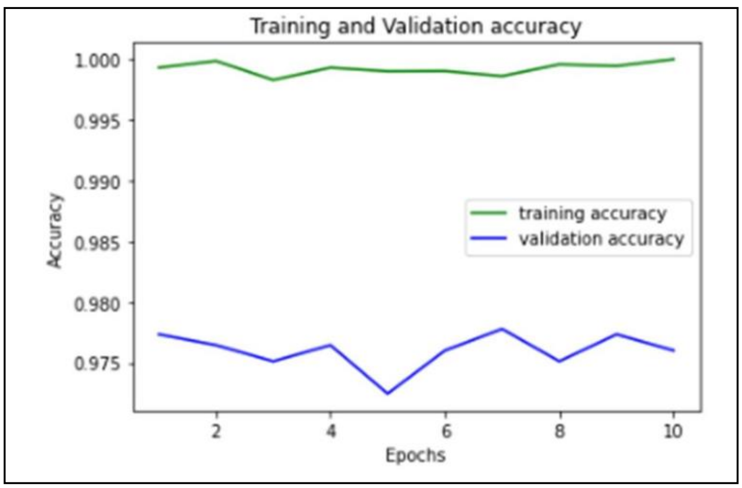
```
Confusion Matrix
[[38 31 33 26 29 22 31 19 21]
 [31 28 25 27 26 26 33 26 28]
 [22 18 28 34 30 36 33 21 28]
 [32 21 23 34 30 24 42 22 22]
 [29 23 29 18 25 30 32 30 34]
 [20 29 27 26 32 25 32 22 37]
 [27 30 26 32 21 31 33 26 24]
 [26 41 25 26 24 26 30 25 27]
 [25 29 33 28 33 30 29 14 29]]

Classification Report
              precision    recall  f1-score   support

           A       0.15      0.15      0.15       250
           B       0.11      0.11      0.11       250
           C       0.11      0.11      0.11       250
           D       0.14      0.14      0.14       250
           E       0.10      0.10      0.10       250
           F       0.10      0.10      0.10       250
           G       0.11      0.13      0.12       250
           H       0.12      0.10      0.11       250
           I       0.12      0.12      0.12       250

    accuracy                           0.12      2250
   macro avg       0.12      0.12      0.12      2250
weighted avg       0.12      0.12      0.12      2250
```

- Accuracy



10. ADVANTAGES & DISADVANTAGES

- Advantages
  - The application is conveniently simple for the end user.
  - The user interface is not complex.
- Disadvantages
  - The dataset in limited.
  - The alphabets only range from 'A' to 'I'.
  - As of now, only static gestures are converted.

11. CONCLUSION

The main objective of this project is to develop gesture recognition so that the deaf can communicate with normal individuals. One of the crucial tasks is the extraction of features, and various gestures should yield various, effectively distinguishable characteristics. To identify the character from the gesture images, we used a trained dataset for the CNN algorithm. These features combined with a labelled data enable accurate real-time ASL alphabet recognition. Our analysis found that accuracy is influenced by a variety of elements, including the camera, dataset, and approach. The accuracy drastically declines in low light and noisy backgrounds.

12. FUTURE SCOPE

The proposed system can be translated into multiple languages, enhancing its dependability and effectiveness. In the near future, it might only be accessible through mobile devices, making the system more convenient and portable. This system is unable to detect gestures made with both hands. Therefore, detecting gestures done with both hands could be a future project.

13. APPENDIX

13.2. SOURCE CODE

- Source Code is available in the GitHub link provided in Section 13.2.

13.3. GITHUB & PROJECT DEMO LINK

- GitHub: https://github.com/IBM-EPBL/IBM-Project-35355-1660284117
- Project Demo Link: https://www.loom.com/share/94a775830ccd410eac69533aa55a4d9c