

Real-Time Communication System Powered by_ **AI for Specially Abled**



TEAM ID: PNT2022TMID00980

1. TEAM LEADER : Dhivagar T
2. TEAM MEMBER1: Balaji G
3. TEAM MEMBER2: Dinesh kumar K
4. TEAM MEMBER3: Dharanidharan D
5. TEAM MEMBER4: Abubakkar Siddiq M

CONTENTS

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

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

7.1 Feature 1

7.2 Feature 2

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code , GITHUB Link & Demo Link

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. Real-time communications (RTC) is any mode of telecommunications in which all users can exchange information instantly.

Communication plays a significant role in making the world better place. It creates a bonding and relations among the people.

1.2 Purpose

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 using the convolutional neural network.

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

2.LITERATURE SURVEY

2.1 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.

2.2 References

- Upendran, S., and Thamizharasi, A., "American Sign Language interpreter system for deaf and dumb individuals", In the Proceedings of the International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 1477-1481, 2014
- Lotti, F., Tiezzi, P., Vassura, G., Biagiotti, L., and Melchiorri, C., "UBH 3: an anthropomorphic hand with simplified endo-skeletal structure and soft continuous fingerpads", In Proceedings IEEE International Conference on Robotics and Automation, 2004 (ICRA'04), Vol.5, pp. 4736-474, IEEE, 2004.
- Rajamohan, A., Hemavathy, R., and Dhanalakshmi, M., "Deaf-Mute Communication Interpreter", International Journal of Scientific Engineering and Technology, Vol.2, No.5, pp.336-341, 2013.
- Verma, P., Shimi S. L. and Priyadarshani, R., "Design of Communication Interpreter for Deaf and Dumb Person", Vol.4, no.1, 2013.

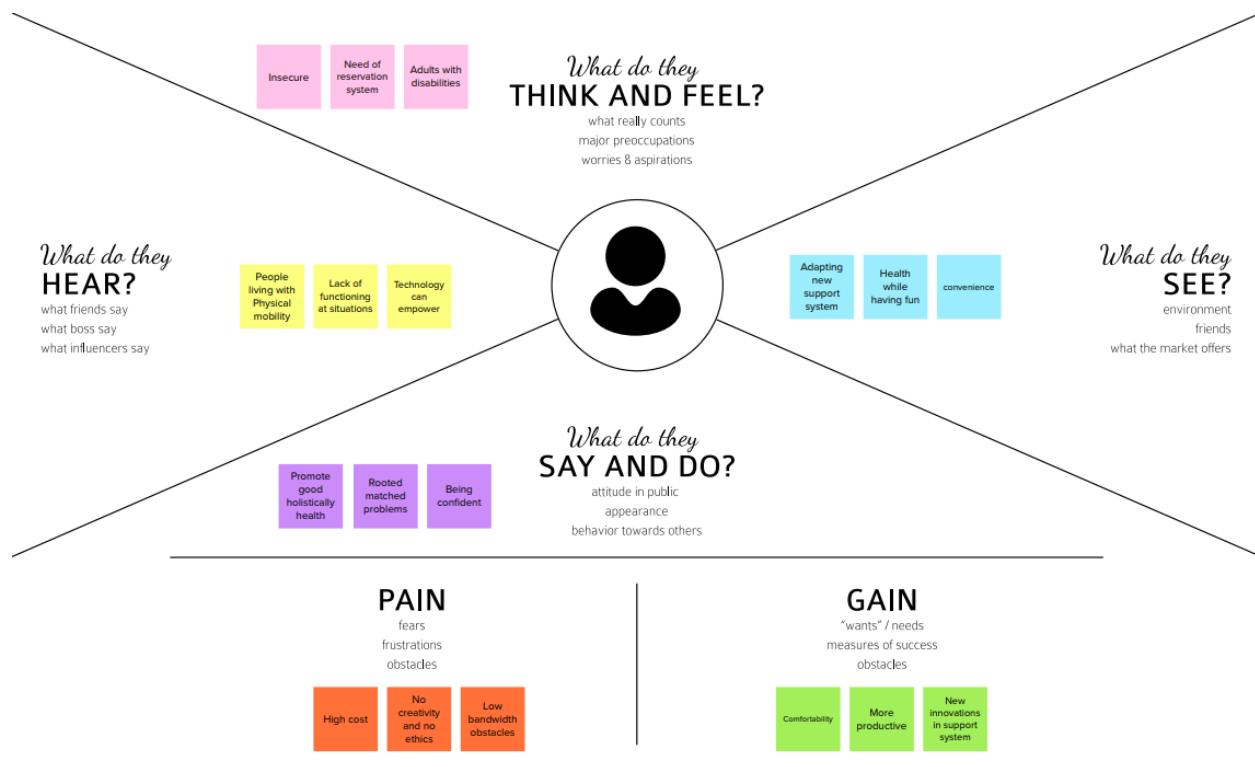
2.3 Problem Statement Definition

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

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

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!

Dhivagar

Analytics	Easy to learn	Ad free
Frictionless navigation feature	Encourages User Engagement	Problem solving, Versatility
Color Scheme	Periodic Update	Free to use

Balaji

Good Image Resolution	Error Correction	Facilitate Communication
Offline work	Security and user privacy	Detailed Information
Open to suggestion	Fast Loading time	Responsive

Dinesh kumar

Show numbers of result in advance	Accurate Updates	Make Distinctive
Understanding User Insight	Consistent Color Scheme	Automatic sort of results
Short Sentences	Great UI and Friendly layout	Actualization

--Dharanidharan

Personalisation	precise and clear	Offline work
Responsive	make distinctive	encourages User Engagement
Push Notification	Detailed one	Expedious

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 can break it up into smaller sub-groups.

🕒 20 minutes

3.3Proposed Solution

Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Deaf and dumb people couldn't able to communicate with the normal people easily.
2.	Idea/Solution description	A real time ML based system is built for the real time sign language detection with a Tensor Flow object detection.
3.	Novelty/Uniqueness	This model using SSD ML algorithm recognizing the signs as words instead of old traditional translators, that are very slow and take too much since every alphabet as to be recognized to form the whole statement in old methods.
4.	Social Impact/Customer satisfaction	It drastically reduce communication difference gap between normal people and specially abled people with the help of AI. So they can live their life independently.
5.	Business Model (RevenueModel)	We use freemium business revenue model for making revenue. In our device, we give most of the basic features for free of charge but they have to pay if they need more advanced features.

6.	Scalability of the Solution	<p>The model which is TensorFlow model that has been used can be replaced with another model as well.</p> <p>The same system can be implemented for different sign languages by substituting the dataset.</p>
----	-----------------------------	---

3.4 Problem Solution fit

Problem-Solution fit canvas 2.0

Purpose / Vision	
1. CUSTOMER SEGMENT(S) CS Who is your customer? People who lost their speech or hearing ability by birth or due to some other factors.	6. CUSTOMER CC What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices. Difficult accessibility, not user friendly, need more technical knowledge to handle, cost,...etc. There are so many choice of solutions available but due to these some constraints, choice of solutions were limited.
2. JOBS-TO-BE-DONE / PROBLEMS J&P What jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides. Deaf and dumb people couldn't able to convey their messages to the normal people easily. Deaf people cannot hear the words as others speaks and dumb people cannot express their feelings by words.	9. PROBLEM ROOT CAUSE RC What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations. In Previously developed solution, they have to use coloured hand gloves for hand position recognition. Also, the old method uses traditional translators which take too much of time to process.
3. TRIGGERS TR What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news. By comparing normal people, Specially Abled people should depend on others and want to live their life independently like other people 4. EMOTIONS: BEFORE / AFTER EM How do customers feel when they face a problem or a job and afterwards? i.e. lost insurance, no confidence in control, loss of trust, communication, obstacles & actions. BEFORE: It is very difficult to convey the message to normal people. AFTER: They overcome their reluctance to have communication with normal people.	5. AVAILABLE SOLUTIONS AS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking. The first ever approach to sign language it has only 6 sign gestures detection. Using colored hands for hand position recognition. But our model is trained to detect different sign languages without any colour gloves, using bare hands only.
8. CHANNELS of BEHAVIOUR CH 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7 Advertise on online with influencers to test the product and promote it also on blog channels 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. On offline, we have our product experience stores where our customer can experience the product in real	7. BEHAVIOUR BE What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace) In our device, there's an option called problem detection display in which our customer can able to see the type of problem occurs & solution will be displayed.

4.REQUIREMENT ANALYSIS

4.1 Functional Requirements

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	User Communication	Communication can be done through pc or mobile camera.
FR-4	User requirement	Option should be shown for hand sign to text and voice conversion and vice versa.
FR-5	Communication requirement	Tutor can be made available to have one to one teaching for user.
FR-6	Regulatory requirements	App shutdown in case of cyber attack
FR-7	Reporting	If any issues found in the application, automatically it will be notified to the developer.
FR-8	Compliance to rules or law	Terms and conditions, private policy, End user subscription agreement.

4.2 Non-Functional requirements

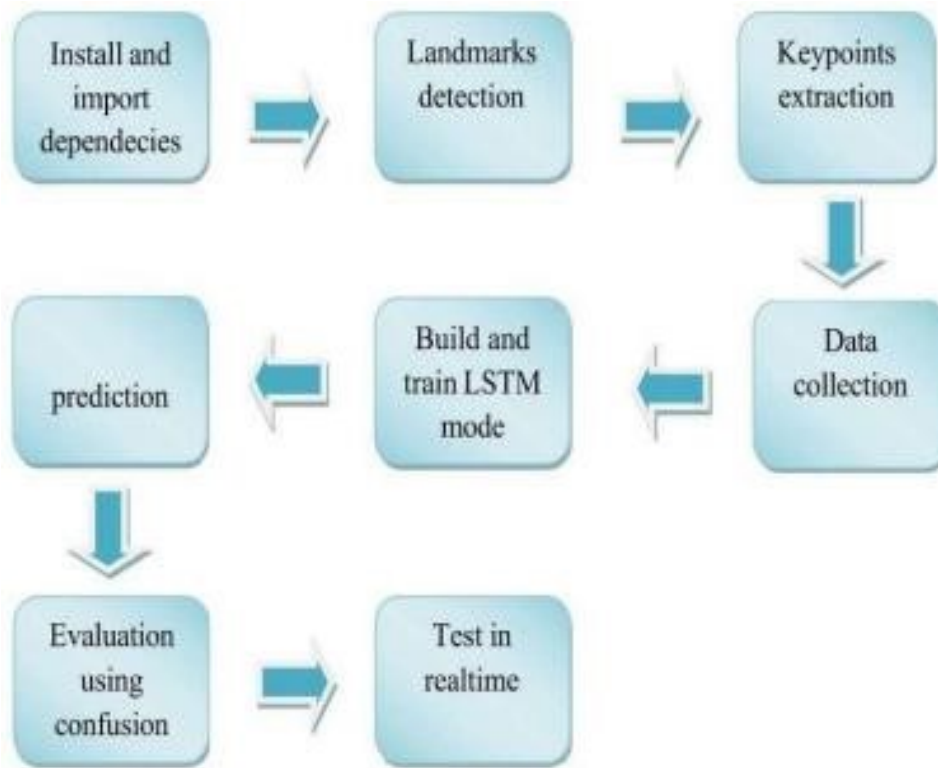
Non-functional Requirements:

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

FR No.	Non-Functional Requirements	Description
NFR-1	Usability	The camera captures all expressions including facial expressions and hand gestures which can be easily used by all age groups. It can be used by deaf-mute people and their care takers.
NFR-2	Security	The system is more secure and information of the customers is also maintained confidentially.
NFR-3	Reliability	The system is very liable, it can last for long amounts of time if well maintained.
NFR-4	Performance	The performance of the model is efficient. The cost-effective nature of the system makes it extremely liable. The latency is very less for the conversion process.
NFR-5	Availability	The solution is suitable for different languages and can be used in many countries. It can be trained for all the available sign languages. This model can be used at any time anywhere.
NFR-6	Scalability	The system gives output rapidly. It also predicts quickly when it gets so many inputs at a time. It predicts different types of sign language at a time. Upto 25000 users can be use this model at a time.

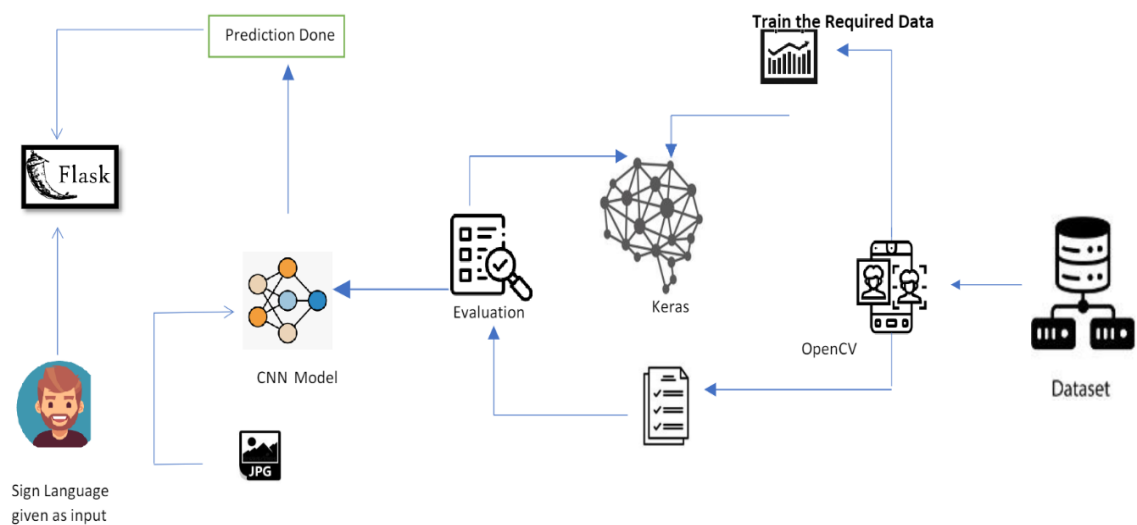
5.PROJECT DESIGN

5.1 Data Flow Diagram



5.2 Solution & Technical Architecture

Technology Architecture



5.3 User Stories

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Data Collection	USN-1	Collect the dataset from alphabet A to Z.	2	Medium
Sprint-1	Training Dataset	USN-2	Train the collected dataset to identify the alphabet	3	High
Sprint-2	Testing the trained model	USN-3	To check whether the data got trained we do the testing for the trained model	1	low
Sprint-2	Saving the text	USN-4	Capture each alphabet and form it as a text and saving it	2	Medium
Sprint-3	Building application	USN-5	Build the flask application and Html page	3	High
Sprint-4	Integrate flask with test code	USN-6	Integrate the flask application with the test code	2	Medium
Sprint-4	Convert text to speech	USN-7	After capturing the text in the application convert to speech	3	High

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Milestone Activity Plan.

Milestone	Function (Epic)	Milestone Story Number	Story / Task
Milestone 1	Data collection	M1	we're collecting dataset for building our project and creating two folders, one for training and another one for testing.
Milestone 2	Image preprocessing	M2	Importing image data generator libraries and applying image data generator functionality to train the test set.
Milestone 3	Model building	M3	Importing the model building libraries, Initializing the model, Adding Convolution layers, Adding the Pooling layers, Adding the Flatten layers, Adding Dense layers, Compiling the model Fit and Save the model.
Milestone 4	Testing the model	M4	Import the packages first. Then we save the model and Load the test image, preprocess it and predict it.
Milestone 5	Application layer	M5	Build the flask application and the HTML pages.
Milestone 6	Train CNN model	M6	Register for IBM Cloud and train Image Classification Model.
Milestone 7	Final result	M7	To ensure all the activities and resulting the final output.

6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	Collect the dataset from alphabet A to Z.	2	Medium	Harish Kumar
Sprint-1	Training Dataset	USN-2	Train the collected dataset to identify the alphabet	3	High	Stanley Deivanayagam N
Sprint-2	Testing the trained model	USN-3	To check whether the data got trained we do the testing for the trained model	1	low	Sachin Prakash Raj. R
Sprint-2	Saving the text	USN-4	Capture each alphabet and form it as a text and saving it	2	Medium	Sajith. M
Sprint-3	Building application	USN-5	Build the flask application and Html page	3	High	Sajith. M
Sprint-4	Integrate flask with test code	USN-6	Integrate the flask application with the test code	2	Medium	.Sachin Prakash Raj. R, Harish Kumar
Sprint-4	Convert text to speech	USN-7	After capturing the text in the application convert to speech	3	High	Sajith. M, Stanley Deivanayagam N

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

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

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
[ ] ls
```

drive/ sample_data/

```
[ ] cd/content/drive/MyDrive/nalaiyathiran
```

/content/drive/MyDrive/nalaiyathiran

```
[ ] ls
```

A22data/ A22data.h5 A22data.zip

```
[ ] pwd
```

'/content/drive/MyDrive/nalaiyathiran'

```
[ ] !unzip A22data.zip
```

Streaming output truncated to the last 5000 lines.

inflating: A22data/training/H/Image_1666640068.7893028.jpg
inflating: A22data/training/H/Image_1666640069.0229783.jpg
inflating: A22data/training/H/Image_1666640069.2740915.jpg
inflating: A22data/training/H/Image_1666640069.4986484.jpg
inflating: A22data/training/H/Image_1666640069.732095.jpg

```
[ ] inflating: A2zdata/training/I/Image_1666640123.3193011.jpg
inflating: A2zdata/training/I/Image_1666640123.6062934.jpg
inflating: A2zdata/training/I/Image_1666640123.9316785.jpg
inflating: A2zdata/training/I/Image_1666640124.285531.jpg
inflating: A2zdata/training/I/Image_1666640124.621861.jpg
inflating: A2zdata/training/I/Image_1666640124.9734807.jpg
inflating: A2zdata/training/I/Image_1666640125.2637064.jpg
inflating: A2zdata/training/I/Image_1666640127.0510988.jpg
inflating: A2zdata/training/I/Image_1666640127.4718907.jpg
inflating: A2zdata/training/I/Image_1666640127.7670643.jpg
inflating: A2zdata/training/I/Image_1666640128.3012989.jpg
inflating: A2zdata/training/I/Image_1666640129.3945134.jpg
inflating: A2zdata/training/I/Image_1666640129.7399273.jpg
inflating: A2zdata/training/I/Image_1666640131.0610347.jpg
inflating: A2zdata/training/I/Image_1666640131.5184145.jpg
```

Image Augmentation

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

```
[ ] train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.2, horizontal_flip=True, vertical_flip=False)
```

```
[ ] test_datagen = ImageDataGenerator(rescale=1./255)
```

```
[ ] x_train = train_datagen.flow_from_directory(r"/content/drive/MyDrive/nalaiyathiran/A2zdata/training", target_size=(100,100), class_mode='categorical', batch_size=75)
```

Found 7132 images belonging to 26 classes.

```
[ ] x_test = test_datagen.flow_from_directory(r"/content/drive/MyDrive/nalaiyathiran/A2zdata/testing", target_size=(100,100), class_mode='categorical', batch_size=75)
```

Found 2862 images belonging to 26 classes.

```
[ ] x_train.class_indices
```

```
{'A': 0,
 'B': 1,
 'C': 2,
 'D': 3,
 'E': 4,
 'F': 5,
 'G': 6,
 'H': 7,
 'I': 8,
 'J': 9,
 'K': 10,
 'L': 11,
 'M': 12,
 'N': 13,
 'O': 14,
 'P': 15,
 'Q': 16,
 'R': 17,
 'S': 18,
 'T': 19,
 'U': 20,
 'V': 21,
 'W': 22,
 'X': 23,
 'Y': 24,
 'Z': 25}
```

Model

```
[ ] from tensorflow.keras.models import Sequential
```

Layers

Layers

```
[ ] from tensorflow.keras.layers import Dense, Convolution2D, MaxPooling2D, Flatten
[ ] model = Sequential()
[ ] model.add(Convolution2D(32, (3,3), input_shape=(100,100,3),activation = 'relu')) #Feature map
[ ] model.add(MaxPooling2D(pool_size = (2,2))) #Pooled matrix
[ ] model.add(Flatten())
[ ] model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 98, 98, 32)	896
max_pooling2d (MaxPooling2D)	(None, 49, 49, 32)	0
flatten (Flatten)	(None, 76832)	0
Total params: 896		
Trainable params: 896		
Non-trainable params: 0		

```
[ ] model.add(Dense(512,activation='relu'))
    model.add(Dense(256,activation='relu'))
[ ] model.add(Dense(26,activation='softmax'))
```

Compile

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

```
[ ] len(x_train)
96
```

```
len(x_test)
```

39

Fit the Model

```
[ ] model.fit_generator(x_train, steps_per_epoch=len(x_train), validation_data=x_test, validation_steps=len(x_test),epochs=5)

/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
    """Entry point for launching an IPython kernel.
Epoch 1/5
96/96 [=====] - 168s 2s/step - loss: 2.4762 - accuracy: 0.5006 - val_loss: 0.7348 - val_accuracy: 0.7628
Epoch 2/5
96/96 [=====] - 147s 2s/step - loss: 0.3840 - accuracy: 0.8850 - val_loss: 0.4430 - val_accuracy: 0.9113
Epoch 3/5
96/96 [=====] - 147s 2s/step - loss: 0.1977 - accuracy: 0.9397 - val_loss: 0.4319 - val_accuracy: 0.9238
Epoch 4/5
```

```
Epoch 4/5
96/96 [=====] - 145s 2s/step - loss: 0.1413 - accuracy: 0.9586 - val_loss: 0.3511 - val_accuracy: 0.9382
Epoch 5/5
96/96 [=====] - 149s 2s/step - loss: 0.0954 - accuracy: 0.9700 - val_loss: 0.2950 - val_accuracy: 0.9486
<keras.callbacks.History at 0x7fce93420890>
```

Save the model

```
[ ] model.save('A2Z.h5')
```

```
[ ] ls
```

```
A2Zdata/ A2Zdata.h5 A2Zdata.zip A2Z.h5
```

Test the model

```
[ ] import numpy as np
    from tensorflow.keras.models import load_model
```

```
[ ] from tensorflow.keras.preprocessing import image
```

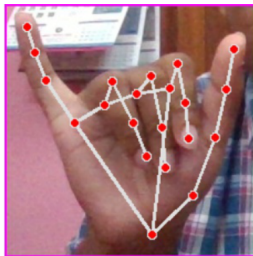
```
[ ] model=load_model('A2Z.h5')
```

```
[ ] pwd
```

```
'/content/drive/MyDrive/nalaiyathiran'
```

```
[ ] img=image.load_img(r'/content/drive/MyDrive/nalaiyathiran/A2Zdata/testing/V/Image_1667328891.1069646.jpg')
```

```
[ ] img
```



```
[ ] img=image.load_img(r'/content/drive/MyDrive/nalaiyathiran/A2Zdata/testing/V/Image_1667328891.1069646.jpg',target_size=(100,100))
```

```
[ ] img
```



```
[ ] x=image.img_to_array(img)
```

```
[ ] x
```

```
[ ] x
```

```
array([[[196., 40., 201.],
        [247., 7., 238.],
        [240., 14., 235.],
        ...,
        [238., 13., 231.],
        [234., 16., 224.],
        [249., 6., 248.]],

       [[208., 34., 207.],
        [243., 132., 200.],
        [219., 146., 175.],
        ...,
        [144., 99., 102.],
        [143., 110., 91.],
        [204., 56., 178.]],

       [[202., 36., 206.],
        [241., 144., 195.],
        [224., 150., 185.],
        ...,
        [149., 110., 81.],
        [148., 110., 87.],
        [205., 58., 173.]],

       ...,

       [[203., 37., 199.],
        [207., 118., 176.],
        [192., 131., 149.],
        ...,
        [116., 133., 141.],
        [138., 166., 170.],
        [209., 93., 226.]],

       [[206., 32., 207.],
        [216., 118., 179.],
        [186., 128., 140.]]])
```

```
[ ]      ...,
        [220., 82., 229.],
        [237., 118., 246.],
        [255., 62., 255.]], dtype=float32)
```

```
[ ] x.shape
```

```
(100, 100, 3)
```

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

```
[ ] x
```

```
array([[[[196., 40., 201.],
        [247., 7., 238.],
        [240., 14., 235.],
        ...,
        [238., 13., 231.],
        [234., 16., 224.],
        [249., 6., 248.]],

       [[208., 34., 207.],
        [243., 132., 200.],
        [219., 146., 175.],
        ...,
        [144., 99., 102.],
        [143., 110., 91.],
        [204., 56., 178.]],

       [[202., 36., 206.],
        [241., 144., 195.],
        [224., 150., 185.],
        ...,
        [149., 110., 81.],
        [148., 110., 87.],
        [205., 58., 173.]]]]])
```



```
[ ]      [238.,  62., 222.],
          [239.,  59., 220.],
          ...,
          [220.,  82., 229.],
          [237., 118., 246.],
          [255.,  62., 255.]]], dtype=float32)
```

```
[ ] x.shape
```

```
(1, 100, 100, 3)
```

```
[ ] y= np.argmax(model.predict(x),axis=1)
```

```
1/1 [=====] - 0s 66ms/step
```

```
[ ] y
```

```
array([24])
```

```
[ ] x_train.class_indices
```

```
{'A': 0,
 'B': 1,
 'C': 2,
 'D': 3,
 'E': 4,
 'F': 5,
 'G': 6,
 'H': 7,
 'I': 8,
 'J': 9,
 'K': 10,
 'L': 11,
 'M': 12,
 'N': 13,
```

```
▶ x_train.class_indices
```

```
{'A': 0,
 'B': 1,
 'C': 2,
 'D': 3,
 'E': 4,
 'F': 5,
 'G': 6,
 'H': 7,
 'I': 8,
 'J': 9,
 'K': 10,
 'L': 11,
 'M': 12,
 'N': 13,
 'O': 14,
 'P': 15,
 'Q': 16,
 'R': 17,
 'S': 18,
 'T': 19,
 'U': 20,
 'V': 21,
 'W': 22,
 'X': 23,
 'Y': 24,
 'Z': 25}
```

```
[ ] index=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
```

```
[ ] index[y[0]]
```

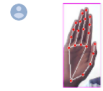
```
'Y'
```

8.TESTING

8.1 Test Cases

```
[ ] img=image.load_img(r'/content/drive/MyDrive/nalaiyathiran/A2Zdata/training/B/Image_1666335447.728338.jpg',target_size=(100,100))
```

```
img
```



```
[ ] x=image.img_to_array(img)
```

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

```
[ ] y=np.argmax(model.predict(x),axis=1)
```

```
1/1 [=====] - 0s 56ms/step
```

```
[ ] y
```

```
array([1])
```

```
[ ] index=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
```

```
[ ] index[y[0]]
```

```
'B'
```

8.2 User Acceptance Testing

1. Purpose of Document

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

2. 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

3. Test Case Analysis

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

9.1 Performance Metrics

Technical Skills Evaluation Matrix									
ID No.	Module	Weightage (%)	Description	Sub-Evaluation Metric & Scoring Criteria	Score	Description	Standard Threshold	Rating	Completion
1	Technical Training & Assignments	10%	This module will be assessed and scored based on the following: 1. Theoretical knowledge in the field of computer science. 2. Application of computer assignments.	Number of Topics / Modules Mastered (Final - 4 Rows)	1	Basic skills of MS Access 2010 - 10 questions	100%	Excellent (Score: 10)	Completed - 1. Passed 1. Not Submitted - 0
				Assignment 1: Database Management	2	Database Management	100%	Completed - 1. Passed 1. Not Submitted - 0	Completed - 1. Passed 1. Not Submitted - 0
				Assignment 2: Database Management	3	Database Management	100%	Completed - 1. Passed 1. Not Submitted - 0	Completed - 1. Passed 1. Not Submitted - 0
2	Innovation & Problem Solving	10%	This module will be assessed and scored based on the following: 1. Innovation in the field of computer science. 2. Application of computer assignments. 3. Innovation in the field of computer science.	1. Diagnostic & Troubleshooting	1	System assessment after issues on the 3rd, 4th, 5th, 6th, 7th, 8th, 9th, 10th, 11th, 12th, 13th, 14th, 15th, 16th, 17th, 18th, 19th, 20th, 21st, 22nd, 23rd, 24th, 25th, 26th, 27th, 28th, 29th, 30th, 31st, 32nd, 33rd, 34th, 35th, 36th, 37th, 38th, 39th, 40th, 41st, 42nd, 43rd, 44th, 45th, 46th, 47th, 48th, 49th, 50th, 51st, 52nd, 53rd, 54th, 55th, 56th, 57th, 58th, 59th, 60th, 61st, 62nd, 63rd, 64th, 65th, 66th, 67th, 68th, 69th, 70th, 71st, 72nd, 73rd, 74th, 75th, 76th, 77th, 78th, 79th, 80th, 81st, 82nd, 83rd, 84th, 85th, 86th, 87th, 88th, 89th, 90th, 91st, 92nd, 93rd, 94th, 95th, 96th, 97th, 98th, 99th, 100th, 101st, 102nd, 103rd, 104th, 105th, 106th, 107th, 108th, 109th, 110th, 111st, 112nd, 113th, 114th, 115th, 116th, 117th, 118th, 119th, 120th, 121st, 122nd, 123rd, 124th, 125th, 126th, 127th, 128th, 129th, 130th, 131st, 132nd, 133rd, 134th, 135th, 136th, 137th, 138th, 139th, 140th, 141st, 142nd, 143rd, 144th, 145th, 146th, 147th, 148th, 149th, 150th, 151st, 152nd, 153rd, 154th, 155th, 156th, 157th, 158th, 159th, 160th, 161st, 162nd, 163rd, 164th, 165th, 166th, 167th, 168th, 169th, 170th, 171st, 172nd, 173rd, 174th, 175th, 176th, 177th, 178th, 179th, 180th, 181st, 182nd, 183rd, 184th, 185th, 186th, 187th, 188th, 189th, 190th, 191st, 192nd, 193rd, 194th, 195th, 196th, 197th, 198th, 199th, 200th, 201st, 202nd, 203rd, 204th, 205th, 206th, 207th, 208th, 209th, 210th, 211st, 212nd, 213th, 214th, 215th, 216th, 217th, 218th, 219th, 220th, 221st, 222nd, 223rd, 224th, 225th, 226th, 227th, 228th, 229th, 230th, 231st, 232nd, 233rd, 234th, 235th, 236th, 237th, 238th, 239th, 240th, 241st, 242nd, 243rd, 244th, 245th, 246th, 247th, 248th, 249th, 250th, 251st, 252nd, 253rd, 254th, 255th, 256th, 257th, 258th, 259th, 260th, 261st, 262nd, 263rd, 264th, 265th, 266th, 267th, 268th, 269th, 270th, 271st, 272nd, 273rd, 274th, 275th, 276th, 277th, 278th, 279th, 280th, 281st, 282nd, 283rd, 284th, 285th, 286th, 287th, 288th, 289th, 290th, 291st, 292nd, 293rd, 294th, 295th, 296th, 297th, 298th, 299th, 300th, 301st, 302nd, 303rd, 304th, 305th, 306th, 307th, 308th, 309th, 310th, 311st, 312nd, 313th, 314th, 315th, 316th, 317th, 318th, 319th, 320th, 321st, 322nd, 323rd, 324th, 325th, 326th, 327th, 328th, 329th, 330th, 331st, 332nd, 333rd, 334th, 335th, 336th, 337th, 338th, 339th, 340th, 341st, 342nd, 343rd, 344th, 345th, 346th, 347th, 348th, 349th, 350th, 351st, 352nd, 353rd, 354th, 355th, 356th, 357th, 358th, 359th, 360th, 361st, 362nd, 363rd, 364th, 365th, 366th, 367th, 368th, 369th, 370th, 371st, 372nd, 373rd, 374th, 375th, 376th, 377th, 378th, 379th, 380th, 381st, 382nd, 383rd, 384th, 385th, 386th, 387th, 388th, 389th, 390th, 391st, 392nd, 393rd, 394th, 395th, 396th, 397th, 398th, 399th, 400th, 401st, 402nd, 403rd, 404th, 405th, 406th, 407th, 408th, 409th, 410th, 411st, 412nd, 413th, 414th, 415th, 416th, 417th, 418th, 419th, 420th, 421st, 422nd, 423rd, 424th, 425th, 426th, 427th, 428th, 429th, 430th, 431st, 432nd, 433rd, 434th, 435th, 436th, 437th, 438th, 439th, 440th, 441st, 442nd, 443rd, 444th, 445th, 446th, 447th, 448th, 449th, 450th, 451st, 452nd, 453rd, 454th, 455th, 456th, 457th, 458th, 459th, 460th, 461st, 462nd, 463rd, 464th, 465th, 466th, 467th, 468th, 469th, 470th, 471st, 472nd, 473rd, 474th, 475th, 476th, 477th, 478th, 479th, 480th, 481st, 482nd, 483rd, 484th, 485th, 486th, 487th, 488th, 489th, 490th, 491st, 492nd, 493rd, 494th, 495th, 496th, 497th, 498th, 499th, 500th, 501st, 502nd, 503rd, 504th, 505th, 506th, 507th, 508th, 509th, 510th, 511st, 512nd, 513th, 514th, 515th, 516th, 517th, 518th, 519th, 520th, 521st, 522nd, 523rd, 524th, 525th, 526th, 527th, 528th, 529th, 530th, 531st, 532nd, 533rd, 534th, 535th, 536th, 537th, 538th, 539th, 540th, 541st, 542nd, 543rd, 544th, 545th, 546th, 547th, 548th, 549th, 550th, 551st, 552nd, 553rd, 554th, 555th, 556th, 557th, 558th, 559th, 560th, 561st, 562nd, 563rd, 564th, 565th, 566th, 567th, 568th, 569th, 570th, 571st, 572nd, 573rd, 574th, 575th, 576th, 577th, 578th, 579th, 580th, 581st, 582nd, 583rd, 584th, 585th, 586th, 587th, 588th, 589th, 590th, 591st, 592nd, 593rd, 594th, 595th, 596th, 597th, 598th, 599th, 600th, 601st, 602nd, 603rd, 604th, 605th, 606th, 607th, 608th, 609th, 610th, 611st, 612nd, 613th, 614th, 615th, 616th, 617th, 618th, 619th, 620th, 621st, 622nd, 623rd, 624th, 625th, 626th, 627th, 628th, 629th, 630th, 631st, 632nd, 633rd, 634th, 635th, 636th, 637th, 638th, 639th, 640th, 641st, 642nd, 643rd, 644th, 645th, 646th, 647th, 648th, 649th, 650th, 651st, 652nd, 653rd, 654th, 655th, 656th, 657th, 658th, 659th, 660th, 661st, 662nd, 663rd, 664th, 665th, 666th, 667th, 668th, 669th, 670th, 671st, 672nd, 673rd, 674th, 675th, 676th, 677th, 678th, 679th, 680th, 681st, 682nd, 683rd, 684th, 685th, 686th, 687th, 688th, 689th, 690th, 691st, 692nd, 693rd, 694th, 695th, 696th, 697th, 698th, 699th, 700th, 701st, 702nd, 703rd, 704th, 705th, 706th, 707th, 708th, 709th, 710th, 711st, 712nd, 713th, 714th, 715th, 716th, 717th, 718th, 719th, 720th, 721st, 722nd, 723rd, 724th, 725th, 726th, 727th, 728th, 729th, 730th, 731st, 732nd, 733rd, 734th, 735th, 736th, 737th, 738th, 739th, 740th, 741st, 742nd, 743rd, 744th, 745th, 746th, 747th, 748th, 749th, 750th, 751st, 752nd, 753rd, 754th, 755th, 756th, 757th, 758th, 759th, 760th, 761st, 762nd, 763rd, 764th, 765th, 766th, 767th, 768th, 769th, 770th, 771st, 772nd, 773rd, 774th, 775th, 776th, 777th, 778th, 779th, 780th, 781st, 782nd, 783rd, 784th, 785th, 786th, 787th, 788th, 789th, 790th, 791st, 792nd, 793rd, 794th, 795th, 796th, 797th, 798th, 799th, 800th, 801st, 802nd, 803rd, 804th, 805th, 806th, 807th, 808th, 809th, 810th, 811st, 812nd, 813th, 814th, 815th, 816th, 817th, 818th, 819th, 820th, 821st, 822nd, 823rd, 824th, 825th, 826th, 827th, 828th, 829th, 830th, 831st, 832nd, 833rd, 834th, 835th, 836th, 837th, 838th, 839th, 840th, 841st, 842nd, 843rd, 844th, 845th, 846th, 847th, 848th, 849th, 850th, 851st, 852nd, 853rd, 854th, 855th, 856th, 857th, 858th, 859th, 860th, 861st, 862nd, 863rd, 864th, 865th, 866th, 867th, 868th, 869th, 870th, 871st, 872nd, 873rd, 874th, 875th, 876th, 877th, 878th, 879th, 880th, 881st, 882nd, 883rd, 884th, 885th, 886th, 887th, 888th, 889th, 890th, 891st, 892nd, 893rd, 894th, 895th, 896th, 897th, 898th, 899th, 900th, 901st, 902nd, 903rd, 904th, 905th, 906th, 907th, 908th, 909th, 910th, 911st, 912nd, 913th, 914th, 915th, 916th, 917th, 918th, 919th, 920th, 921st, 922nd, 923rd, 924th, 925th, 926th, 927th, 928th, 929th, 930th, 931st, 932nd, 933rd, 934th, 935th, 936th, 937th, 938th, 939th, 940th, 941st, 942nd, 943rd, 944th, 945th, 946th, 947th, 948th, 949th, 950th, 951st, 952nd, 953rd, 954th, 955th, 956th, 957th, 958th, 959th, 960th, 961st, 962nd, 963rd, 964th, 965th, 966th, 967th, 968th, 969th, 970th, 971st, 972nd, 973rd, 974th, 975th, 976th, 977th, 978th, 979th, 980th, 981st, 982nd, 983rd, 984th, 985th, 986th, 987th, 988th, 989th, 990th, 991st, 992nd, 993rd, 994th, 995th, 996th, 997th, 998th, 999th, 1000th, 1001st, 1002nd, 1003rd, 1004th, 1005th, 1006th, 1007th, 1008th, 1009th, 1010th, 1011st, 1012nd, 1013th, 1014th, 1015th, 1016th, 1017th, 1018th, 1019th, 1020th, 1021st, 1022nd, 1023rd, 1024th, 1025th, 1026th, 1027th, 1028th, 1029th, 1030th, 1031st, 1032nd, 1033rd, 1034th, 1035th, 1036th, 1037th, 1038th, 1039th, 1040th, 1041st, 1042nd, 1043rd, 1044th, 1045th, 1046th, 1047th, 1048th, 1049th, 1050th, 1051st, 1052nd, 1053rd, 1054th, 1055th, 1056th, 1057th, 1058th, 1059th, 1060th, 1061st, 1062nd, 1063rd, 1064th, 1065th, 1066th, 1067th, 1068th, 1069th, 1070th, 1071st, 1072nd, 1073rd, 1074th, 1075th, 1076th, 1077th, 1078th, 1079th, 1080th, 1081st, 1082nd, 1083rd, 1084th, 1085th, 1086th, 1087th, 1088th, 1089th, 1090th, 1091st, 1092nd, 1093rd, 1094th, 1095th, 1096th, 1097th, 1098th, 1099th, 1100th, 1101st, 1102nd, 1103rd, 1104th, 1105th, 1106th, 1107th, 1108th, 1109th, 1110th, 1111st, 1112nd, 1113th, 1114th, 1115th, 1116th, 1117th, 1118th, 1119th, 1120th, 1121st, 1122nd, 1123rd, 1124th, 1125th, 1126th, 1127th, 1128th, 1129th, 1130th, 1131st, 1132nd, 1133rd, 1134th, 1135th, 1136th, 1137th, 1138th, 1139th, 1140th, 1141st, 1142nd, 1143rd, 1144th, 1145th, 1146th, 1147th, 1148th, 1149th, 1150th, 1151st, 1152nd, 1153rd, 1154th, 1155th, 1156th, 1157th, 1158th, 1159th, 1160th, 1161st, 1162nd, 1163rd, 1164th, 1165th, 1166th, 1167th, 1168th, 1169th, 1170th, 1171st, 1172nd, 1173rd, 1174th, 1175th, 1176th, 1177th, 1178th, 1179th, 1180th, 1181st, 1182nd, 1183rd, 1184th, 1185th, 1186th, 1187th, 1188th, 1189th, 1190th, 1191st, 1192nd, 1193rd, 1194th, 1195th, 1196th, 1197th, 1198th, 1199th, 1200th, 1201st, 1202nd, 1203rd, 1204th, 1205th, 1206th, 1207th, 1208th, 1209th, 1210th, 1211st, 1212nd, 1213th, 1214th, 1215th, 1216th, 1217th, 1218th, 1219th, 1220th, 1221st, 1222nd, 1223rd, 1224th, 1225th, 1226th, 1227th, 1228th, 1229th, 1230th, 1231st, 1232nd, 1233rd, 1234th, 1235th, 1236th, 1237th, 1238th, 1239th, 1240th, 1241st, 1242nd, 1243rd, 1244th, 1245th, 1246th, 1247th, 1248th, 1249th, 1250th, 1251st, 1252nd, 1253rd, 1254th, 1255th, 1256th, 1257th, 1258th, 1259th, 1260th, 1261st, 1262nd, 1263rd, 1264th, 1265th, 1266th, 1267th, 1268th, 1269th, 1270th, 1271st, 1272nd, 1273rd, 1274th, 1275th, 1276th, 1277th, 1278th, 1279th, 1280th, 1281st, 1282nd, 1283rd, 1284th, 1285th, 1286th, 1287th, 1288th, 1289th, 1290th, 1291st, 1292nd, 1293rd, 1294th, 1295th, 1296th, 1297th, 1298th, 1299th, 1300th, 1301st, 1302nd, 1303rd, 1304th, 1305th, 1306th, 1307th, 1308th, 1309th, 1310th, 1311st, 1312nd, 1313th, 1314th, 1315th, 1316th, 1317th, 1318th, 1319th, 1320th, 1321st, 1322nd, 1323rd, 1324th, 1325th, 1326th, 1327th, 1328th, 1329th, 1330th, 1331st, 1332nd, 1333rd, 1334th, 1335th, 1336th, 1337th, 1338th, 1339th, 1340th, 1341st, 1342nd, 1343rd, 1344th, 1345th, 1346th, 1347th, 1348th, 1349th, 1350th, 1351st, 1352nd, 1353rd, 1354th, 1355th, 1356th, 1357th, 1358th, 1359th, 1360th, 1361st, 1362nd, 1363rd, 1364th, 1365th, 1366th, 1367th, 1368th, 1369th, 1370th, 1371st, 1372nd, 1373rd, 1374th, 1375th, 1376th, 1377th, 1378th, 1379th, 1380th, 1381st, 1382nd, 1383rd, 1384th, 1385th, 1386th, 1387th, 1388th, 1389th, 1390th, 1391st, 1392nd, 1393rd, 1394th, 1395th, 1396th, 1397th, 1398th, 1399th, 1400th, 1401st, 1402nd, 1403rd, 1404th, 1405th, 1406th, 1407th, 1408th, 1409th, 1410th, 1411st, 1412nd, 1413th, 1414th, 1415th, 1416th, 1417th, 1418th, 1419th, 1420th, 1421st, 1422nd, 1423rd, 1424th, 1425th, 1426th, 1427th, 1428th, 1429th, 1430th, 1431st, 1432nd, 1433rd, 1434th, 1435th, 1436th, 1437th, 1438th, 1439th, 1440th, 1441st, 1442nd, 1443rd, 1444th, 1445th, 1446th, 1447th, 1448th, 1449th, 1450th, 1451st, 1452nd, 1453rd, 1454th, 1455th, 1456th, 1457th, 1458th, 1459th, 1460th, 1461st, 1462nd, 1463rd, 1464th, 1465th, 1466th, 1467th, 1468th, 1469th, 1470th, 1471st, 1472nd, 1473rd, 1474th, 1475th, 1476th, 1477th, 1478th, 1479th, 1480th, 1481st, 1482nd, 1483rd, 1484th, 1485th, 1486th, 1487th, 1488th, 1489th, 1490th, 1491st, 1492nd, 1493rd, 1494th, 1495th, 1496th, 1497th, 1498th, 1499th, 1500th, 1501st, 1502nd, 1503rd, 1504th, 1505th, 1506th, 1507th, 1508th, 1509th, 1510th, 1511st, 1512nd, 1513th, 1514th, 1515th, 1516th, 1517th, 1518th, 1519th, 1520th, 1521st, 1522nd, 1523rd, 1524th, 1525th, 1526th, 1527th, 1528th, 1529th, 1530th, 1531st, 1532nd, 1533rd, 1534th, 1535th, 1536th, 1537th, 1538th, 1539th, 1540th, 1541st, 1542nd, 1543rd, 1544th, 1545th, 1546th, 1547th, 1548th, 1549th, 1550th, 1551st, 1552nd, 1553rd, 1554th, 1555th, 1556th, 1557th, 1558th, 1559th, 1560th, 1561st, 1562nd, 1563rd, 1564th, 1565th, 1566th, 1567th, 1568th, 1569th, 1570th, 1571st, 1572nd, 1573rd, 1574th, 1575th, 1576th, 1577th, 1578th, 1579th, 1580th, 1581st, 1582nd, 1583rd, 1584th, 1585th, 1586th, 1587th, 1588th, 1589th, 1590th, 1591st, 1592nd, 1593rd, 1594th, 1595th, 1596th, 1597th, 1598th, 1599th, 1600th, 1601st, 1602nd, 1603rd, 1604th, 1605th, 1606th, 1607th, 1608th, 1609th, 161			

10.ADVANTAGES & DISADVANTAGES

Advantages:

- It is possible to create a mobile application to bridge the communication gap between deaf and dumb persons and the general public.
- As different sign language standards exist, their dataset can be added, and the user can choose which sign language to read.

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 peoples by normal peoples this system is proposed.

12. FUTURE SCOPE

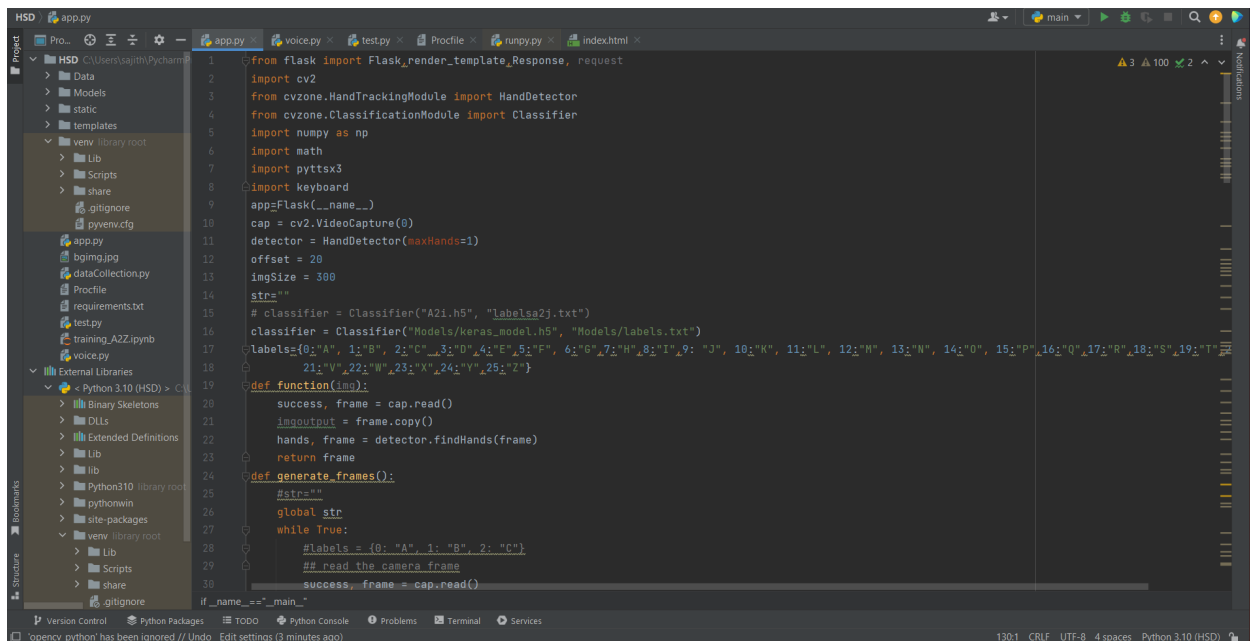
The speech-to-text and text-to-speech technologies helped those people who had difficulties in communicating or expressing their feelings to the normal people.

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 from flask import Flask, render_template, Response, request
2 import cv2
3 from cvzone.HandTrackingModule import HandDetector
4 from cvzone.ClassificationModule import Classifier
5 import numpy as np
6 import math
7 import pyttsx3
8 import keyboard
9 app = Flask(__name__)
10 cap = cv2.VideoCapture(0)
11 detector = HandDetector(maxHands=1)
12 offset = 20
13 imgSize = 300
14 str = ""
15 # classifier = Classifier("A21.h5", "labela2j.txt")
16 classifier = Classifier("Models/keras_model.h5", "Models/Labels.txt")
17 labels = {0: "A", 1: "B", 2: "C", 3: "D", 4: "E", 5: "F", 6: "G", 7: "H", 8: "I", 9: "J", 10: "K", 11: "L", 12: "M", 13: "N", 14: "O", 15: "P", 16: "Q", 17: "R", 18: "S", 19: "T",
18         20: "U", 21: "V", 22: "W", 23: "X", 24: "Y", 25: "Z"}
19 def function(img):
20     success, frame = cap.read()
21     imgOutput = frame.copy()
22     hands, frame = detector.findHands(frame)
23     return frame
24 def generate_frames():
25     # str = ""
26     global str
27     while True:
28         # labels = {0: "A", 1: "B", 2: "C"}
29         # read the camera frame
30         success, frame = cap.read()
31         if __name__ == '__main__':
```

This screenshot shows the first part of the Python script 'app.py' in the PyCharm IDE. The code is as follows:

```
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
generate_frames()
```

The code defines a global string 'str' and enters a 'while True' loop. Inside the loop, it attempts to read a camera frame. If successful, it finds hands using a detector. If hands are found, it takes the first hand, calculates its bounding box, and creates a white image of the hand. It then crops the original frame to the hand's bounding box and resizes it. The cropped and resized image is then used for prediction by a classifier. The prediction index is printed, and the string 'str' is updated with the label corresponding to the prediction index. The loop continues until the 's' key is pressed.

This screenshot shows the second part of the Python script 'app.py' in the PyCharm IDE. The code is as follows:

```
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
generate_frames()
```

The code continues the 'while True' loop. It updates the string 'str' with the label corresponding to the prediction index. It then uses 'cv2.putText' to display the label on the image. If the 'a' key is pressed, it appends the label to the string. If the 'd' key is pressed, it removes the last character from the string. If the 'w' key is pressed, it resets the string. If the 's' key is pressed, it displays the string on the image. The loop continues until the 's' key is pressed.

```
79         if keyboard.is_pressed('a'):
80             str += " "
81             cv2.putText(imgOutput, str, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 3)
82         if keyboard.is_pressed('d'):
83             str = str[:-1]
84             cv2.putText(imgOutput, str, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 3)
85         if keyboard.is_pressed('#'):
86             str=""
87             cv2.putText(imgOutput, str, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 3)
88         cv2.rectangle(imgOutput, (x - offset, y - offset - 50),
89                       (x - offset + 90, y - offset - 50 + 50), (255, 0, 255), cv2.FILLED)
90         cv2.putText(imgOutput, labels[index], (x, y - 26), cv2.FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255), 2)
91         cv2.rectangle(imgOutput, (x - offset, y - offset),
92                       (x + w + offset, y + h + offset), (255, 0, 255), 4)
93         cv2.putText(imgOutput, str, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 3)
94         ret,buffer=cv2.imencode('.jpg',imgOutput)
95         imgOutput=buffer.tobytes()
96         yield(b'--frame\r\n'
97              + b'Content-Type: image/jpeg\r\n\r\n' + imgOutput + b'\r\n')
98         return render_template("index.html", pred=estr)
99         @app.route('/predict',methods=['POST','GET'])
100     def predictions():
101         return render_template("index.html", pred=estr)
102         # return generate_frames()
103         @app.route('/stop',methods=['POST','GET'])
104     def stopping():
105         count = 0
106         while True:
107             ## read the camera frame
108             success_frame=cap.read()
```

```
109             if not success:
110                 return "The text is converted into voice.Restart the app again to start predicting.Thank you!!!!!!!"
111                 break
112             # if count==1:
113             #     return "Exceeded"
114             #     break
115             else:
116                 #cap.release()
117                 #print("The Recorded String is:", str)
118                 text2speech = pyttsx3.init()
119                 newVoiceRate = 125
120                 text2speech.setProperty('rate', newVoiceRate)
121                 text2speech.say(str)
122                 text2speech.runAndWait()
123                 return render_template('index.html')
124         @app.route('/')
125     def index():
126         return render_template("index.html")
127     stopping()
128     while True
```


The screenshot shows a PyCharm IDE with a project named 'HSD'. The file explorer on the left shows a directory structure with files like 'app.py', 'bgimg.jpg', 'dataCollection.py', 'Profile', 'requirements.txt', 'test.py', 'training_A22.ipynb', 'voice.py', and various external libraries. The main editor displays the code for 'app.py', which includes a Flask application with routes for index and video, and a text-to-speech function using pyttsx3. The code is as follows:

```
110     return "The text is converted into voice.Restart the app again to start predicting.Thank you!!!!!!!"
111     break
112     # if count==1:
113     #     return "Exceeded"
114     break
115 else:
116     #cap.release()
117     #print("The Recorded String is:", str)
118     text2speech = pyttsx3.init()
119     newVoiceRate = 125
120     text2speech.setProperty('rate', newVoiceRate)
121     text2speech.say(str)
122     text2speech.runAndWait()
123     return render_template('index.html')
124 @app.route('/')
125 def index():
126     return render_template('index.html')
127 @app.route('/video')
128 def video():
129     return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
130 if __name__ == '__main__':
131     app.run(debug=True)
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

GITHUB LINK:

<https://github.com/IBM-EPBL/IBM-Project-5609-1658811745>