A picture containing dark

Description automatically generated

Gesture AI

Helping Deaf and Dumb

A Machine learning Model

Team #1

December 14th, 2021

Team Members

**Team Member 1**

•Requirements Specifications

•Primary Class List  
•Class Diagrams  
•Use Case Diagrams

•Sequence Diagrams

•GUI Mockups  
•Class Skeletons

•Implementation Plan

•Coding  
•Unit Testing

**Team Member 2**

•Scenarios  
•Class Skeletons

•GUI Mockups   
•Class Skeletons

•Implementation Plan

•Test Cases

**Team Member 3**

•Use cases

•Implementation Plan

•Database testing

•Interface design

•Unit Testing

**Team Member 4**

•Use Case Diagrams

•Primary Class List  
•Class Diagrams

•Sequence Diagrams

•GUI Mockups

•Implementation Plan

•Test Cases

TABLE OF CONTENTS:

1. Summary of Changes..........................................................................5

2. Proposal................................................................................................6

3. Glossary of Terms...............................................................................7

4. System Requirements.....................................................................8-11

5. Primary Class List..............................................................................12

6. Class Diagrams....................................................................................13

7. Use Cases.......................................................................................14-15

8. Use Case Diagrams.......................................................................16-17

9. Schema for Detailed Use Cases...................................................18-28

10. Traceability matrix.................................................................... 29-30

10.Sequence Diagrams............................................................................31

11. Model Skeletons...........................................................................32-35

12.GUI Mockups.................................................................................36-38

13. Implementation Plan........................................................................39

14. Implementation Breakdown............................................................40

15. Implementation Overview……………………………………………………….41

16. Test Cases.....................................................................................42-46

17. State diagram…………………………………………………………………….47-49

17. Problem frames............................................................................50-61

18. Source Code........................................................................................62

SUMMARY OF CHANGES:

-------------------10/11/21-------------------------

~Skipped the face keypoints and created the dataset without it

--------------------9/24/21---------------------

~ Started making the dataset by showing gestures with 30 frames per action.

~ Started to join the Machine learning and the development of the project

~ Increased the accuracy of the model from 70 to 81 %

~ Added some front-end to the model using Flask

~ Started to use mediapipe for better gesture recognition

--------------------9/22/21---------------------

~ Modified the web-app for integration with the machine learning model.

PROJECT PROPOSAL:

There have been several advancements in technology and a lot of research has. been done to help people who are deaf and dumb. Aiding the cause, Deep learning, and computer vision can be used too to make an impact on this cause. This can be very helpful for the deaf and dumb people in communicating with others as knowing sign language is not something that is common to all, moreover, this can be extended to creating automatic editors, where the person can easily write by just their hand gestures

A real-time gesture detection and classification application that helps the deaf or dumb to translate their hand sign language or gesture to audio/text for the public and vice versa.

We are trying to make communication easier and accessible to the dumb and deaf. And provide a tool which will help them in their day-to-day life.

Many restaurants and shopping malls do not have special gesture tools which would help the public. Safer, quicker, and convenient means to help the people by converting sign language to text by making a machine learning algorithm.

GLOSSARY OF TERMS:

User: The one who interacts with the end created application.

Hand gesture database: All the hand gestures that are collected by the user are saved into the database.

User interface: The visual part of the system that the user will detect hand gestures.

LSTM: An artificial recurrent neural network architecture used to detect hand gestures.

Camera: To capture all the hand gestures that the user enters and to capture the dataset properly.

ML model: Collects the captured images and detects the sign language, converts sign language to text/ speech.

System: system will be able to scale the program with high response time and security.

Microphone: used to listen the text audio.

Gesture: To be able to show the gestures to the camera so that it reads the user’s hand correctly.

Audio: to listen to the audio generated from the sign language.

Scale: Able to scale the program with high response time and security.

SYSTEM REQUIREMENTS:

**Functional requirements:**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Priority** | **Requirements** |
| **REQ-1** | **5** | **The program should be able to use the device’s camera to take**  **the key points from the data processed to it.** |
| **REQ-2** | **5** | **The program must have the microphone of the device accessible every**  **time an audio option is selected for giving data to the system via audio**  **source.** |
| **REQ-3** | **4** | **The program must have a user-interface, where the user can use the**  **device camera for showing gestures so that the model takes that data and**  **converts it into text.** |
| **REQ-4** | **1** | **The program’s GUI should also have a textbox, where the user can type in**  **the text which could be converted to gesture language.** |
| **REQ-5** | **2** | **The program should be asking the user whether he wants to allow the**  **software to use the device camera or not** |
| **REQ-6** | **4** | **The program should be predicting the gestures accurately.** |
| **REQ-7** | **1** | **The program should be able to recognize the hand of the user correctly** |
| **REQ-8** | **2** | **The program should be able to use the speaker of the device** |
| **REQ-9** | **1** | **The program should have a drop box for switching from**  **Self-training option and Gesture option** |
| **REQ-10** | **2** | **The program should have an option for choosing where the model**  **currently should be whether at collecting data, training data or testing**  **data** |
| **REQ-11** | **2** | **The program should have a scale where the user can select how many**  **samples would be taken for self-training.** |
| **REQ-12** | **2** | **The program should also have scales for test ratio, layers and**  **neurons to be selected for the current self-training regime being**  **done.** |
| **REQ-13** | **2** | **The program should also have provision to generate audio from the sign**  **language captured for people with low or no eyesight.** |

**Non-functional requirements:**

**non-functional requirements**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Priority** | **Requirements** |
| **Req-1** | **4** | **The program’s response time of the model should be fast** |
| **Req-2** | **3** | **The program’s model should be scalable and be able to handle large amounts of data** |
| **Req-3** | **3** | **The program’s code should be easily manageable and understood by other programmers** |
| **Req-4** | **4** | **The program should be able to provide security to data taken from users.** |
| **Req-5** | **3** | **The program should be able to read each gesture in 1 minute or less than that.** |
| **Req-6** | **5** | **The program should be easily running on low-end devices.** |
| **Req-7** | **3** | **The program should be able to optimize the memory usage of the system.** |

**Enumerated User Interface:**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Priority** | **Requirements** |
| **Req-1** | **5** | **The program’s user-interface must be simple to understand and use** |
| **Req-2** | **4** | **The program should have a consistent theme for all pages.** |
| **Req-3** | **3** | **The program’s interface must be able to adjust according to different devices and different screen sizes.** |

PRIMARY CLASS LIST:

1. User
2. Camera
3. Dataset
4. Bone Structure
5. Keypoints
6. Display
7. Machine learning Model

Class Diagrams

Chart, box and whisker chart

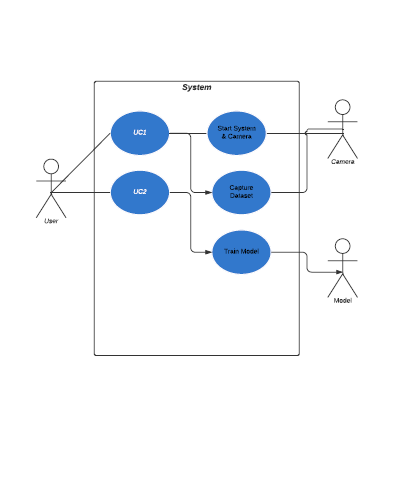
Description automatically generated

**Use Cases from System Requirements:**

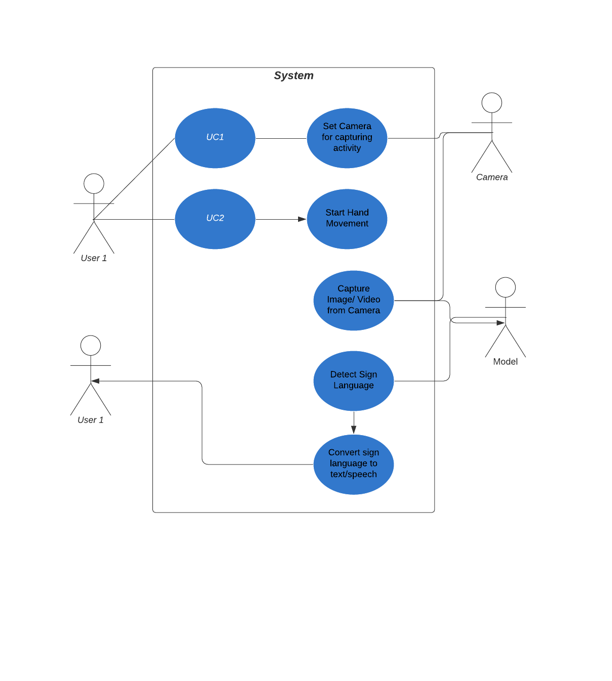
|  |  |  |
| --- | --- | --- |
| **Actor** | **Actor’s Goal (what the actor intends to accomplish)** | **Use case Name** |
| User | To use the camera on which the program is running | Camera (UC-1) |
| User | To use the microphone of the device | Microphone (UC-2) |
| User | To type text in a textbox for converting it to gesture language. | Text (UC-3) |
| User/Camera | To be able to show the gestures to the camera so that it reads the user’s hand correctly | Gesture (UC-4) |
| User | To use the speaker of the device via the program | Speaker (UC -5) |
| User | To navigate between Self-training mode and gesture reading option | Navigate (UC-6) |
| User | To select the size of samples for training | Size (UC-7) |
| User | To listen the audio generated from sign language | Audio (UC-8) |
| System | To be able to scale the program with high response time and security | Scale (UC-9) |
| Camera | To capture the dataset properly | UC-1, UC-6, UC-7 |

USE CASE DIAGRAM:

DATA MODELLING SYSTEM DIAGRAM

****

GESTURE SUBSYSTEM DIAGRAM:

****

**SCHEMA FOR DETAILED USECASE:**

**Use Case 1: Camera**

|  |  |
| --- | --- |
| **Use Case UC-1.** | **Camera** |
| **Related requirements:** | REQ-1, REQ-5, REQ-6 stated in enumerated functional requirements table above |
| **Initiating Actor:** | Any end-user |
| **Actor’s Actor’s Goal:** | To use the camera on which the program is running |
| **Participating Actors:** | Camera, System |
| **Preconditions:** | The camera of the device is not being used by the program and the user is on the welcome page. |
| **Postconditions:** | The user is using the camera easily for showing different kinds of gestures. |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user selects to start the camera.  <- 2. The program uses the camera of the device.  -> 3. The end-user starts to show some gestures.  <- 4. The program reads the hand of user to save the gestures | |

**Use Case 2: Microphone**

|  |  |
| --- | --- |
| **Use Case UC-2.** | **Microphone** |
| **Related requirements:** | REQ-2, REQ-8, REQ-13 stated in enumerated functional requirements table above |
| **Initiating Initiating Actor:** | Any end-user |
| **Actor’s Actor’s Goal:** | To use the microphone of the device. |
| **Participating Actors:** | Microphone, System |
| **Preconditions:** | The microphone of the device is not being used by the program. |
| **Postconditions:** | The user is using the microphone easily for using audio options. |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user selects to use the microphone.  <- 2. The program starts to access the microphone.  -> 3. The end-user starts to record his voice.  <- 4. The program converts the audio to text and gestures. | |

**Use Case 3: Text**

|  |  |
| --- | --- |
| **Use Case UC-1.** | **Text** |
| **Related requirements:** | REQ-4, REQ-12 stated in the enumerated functional requirements table above.  REQ-1 stated in the enumerated user interface table above. |
| **Initiating Initiating Actor:** | Any end-user |
| **Actor’s Actor’s Goal:** | To type text in a textbox for converting it to gesture language. |
| **Participating Actors:** | System. |
| **Preconditions:** | The textbox is empty |
| **Postconditions:** | The textbox is being used for converting text to gestures. |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user types in something in the textbox.  <- 2. The program starts to convert to gestures.  <- 3. The program outputs the converted gestures on screen. | |

**Use Case 4: Gesture**

|  |  |
| --- | --- |
| **Use Case UC-4.** | **Gesture** |
| **Related requirements:** | REQ-3, REQ-4, REQ-6, REQ-7 stated in the enumerated functional requirements table above.  REQ-1 stated in the enumerated user interface table above. |
| **Initiating Initiating Actor:** | Any end-user |
| **Actor’s Actor’s Goal:** | To capture the hand gesture to interpret sign language. |
| **Participating Actors:** | System. |
| **Preconditions:** | The camera should be attached to the system and working |
| **Postconditions:** | The camera is used for converting gestures to text or speech |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user starts the camera of the device and starts doing hand gestures  <- 2. The program reads the hand gestures of the user.  <- 3. The program outputs the text which is derived from the gestures of the user. | |

**Use Case 5: Speaker**

|  |  |
| --- | --- |
| **Use Case UC-5.** | **Speaker** |
| **Related requirements:** | REQ-8 stated in the enumerated functional requirements table above. |
| **Initiating Initiating Actor:** | Any end-user |
| **Actor’s Actor’s Goal:** | To use the speaker of the device via the program |
| **Participating Actors:** | System. |
| **Preconditions:** | The speaker of the device is not being used. |
| **Postconditions:** | The user has used the speakers to output audio. |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user asks for access to speakers of the device.  <- 2. The program allows access.  -> 3. The user outputs the audio through speakers. | |

**Use Case 6: Navigate**

|  |  |
| --- | --- |
| **Use Case UC-6.** | **Navigate** |
| **Related requirements:** | REQ-9, REQ-10 stated in the enumerated functional requirements table above. |
| **Initiating Actor:** | Any end-user |
| **Actor’s Goal:** | To navigate between Self-training mode and gesture reading option |
| **Participating Actors:** | System. |
| **Preconditions:** | Can switch between self-training and gesture detection mode. |
| **Postconditions:** | By selecting the mode, self-training or gesture recognition can be performed. |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user selects from two modes (1. Self-Training 2. Sign Language Detection).  <- 2. The program starts to convert gestures to text or speech.  <- 3. The program outputs the converted gestures on screen in the form of text or speech. | |

**Use Case 7: Size**

|  |  |
| --- | --- |
| **Use Case UC-7.** | **Size** |
| **Related requirements:** | REQ-11 stated in the enumerated functional requirements table above.  REQ-1 stated in the enumerated user interface table above. |
| **Initiating Actor:** | Any end-user |
| **Actor’s Goal:** | To type the size of the training sample. |
| **Participating Actors:** | User. |
| **Preconditions:** | The text input to input size of training sample. |
| **Postconditions:** | The textbox is being used for converting text to gestures. |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user can type the size of the training sample to be captured.  <- 2. The program starts to capture the samples.  <- 3. The program outputs N number of images to be used for training. | |

**Use Case 8: Audio**

|  |  |
| --- | --- |
| **Use Case UC-8.** | **Audio** |
| **Related requirements:** | REQ-2 stated in the enumerated functional requirements table above.  REQ-1 stated in the enumerated user interface table above. |
| **Initiating Actor:** | Any end-user |
| **Actor’s Goal:** | To generate the audio from the sign language gesture captured from the camera. |
| **Participating Actors:** | System, Machine Learning Model, Camera, Speaker. |
| **Preconditions:** | System should be connected to the speaker. |
| **Postconditions:** | The user should get audio for the sign language detected using a camera and machine learning model. |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user performs sign language gestures.  <- 2. The model converts the gestures to text.  <- 3. The output will be the speech which will be converted from text obtained from the model. | |

**Use Case 9: Scale**

|  |  |
| --- | --- |
| **Use Case UC-1.** | **Scale** |
| **Related requirements:** | REQ-12 stated in the enumerated functional requirements table above.  REQ-1, REQ-2, REQ-4 stated in the enumerated non-functional requirements table above. |
| **Initiating Actor:** | Any end-user |
| **Actor’s Goal:** | To be able to scale the program with high response time and security |
| **Participating Actors:** | System, Machine Learning Model, Camera, Speaker. |
| **Preconditions:** | The program isn't scalable to handle more big data. |
| **Postconditions:** | The program is able to handle more big data. |
| **Flow of Events for Main Success Scenario:** | |
| -> 1. The user inputs big data in the program  <- 2. The program is processing that data for the user.  <- 3. The program gives the desired output back to the user. | |

**Traceability matrix:**

**Req 1:** Open camera to capture image.

**Req 2:** Access the microphone.

**Req 3:** Interface for the user to capture images.

**Req 4:** Textbox to type the text.

**Req 5:** Asking whether to use the device camera or not.

**Req 6:** Predict gestures.

**Req 7:** Recognizing the data correctly

**Req 8:** Use speakers.

**Req 9:** Switch from self-training option to gesture tool.

**Req 10:** Choose to collect, train or test data.

**Req 11:** Scale to select samples for scaling

**Req 12:** Scaletest ratio, layers and neurons

**Req 13:** Generate audio from sign language.

**Use cases:**

UC 1: Camera

UC 2: Microphone

UC 3: Text

UC 4: Gesture

UC 5: Speaker

UC 6: Navigate

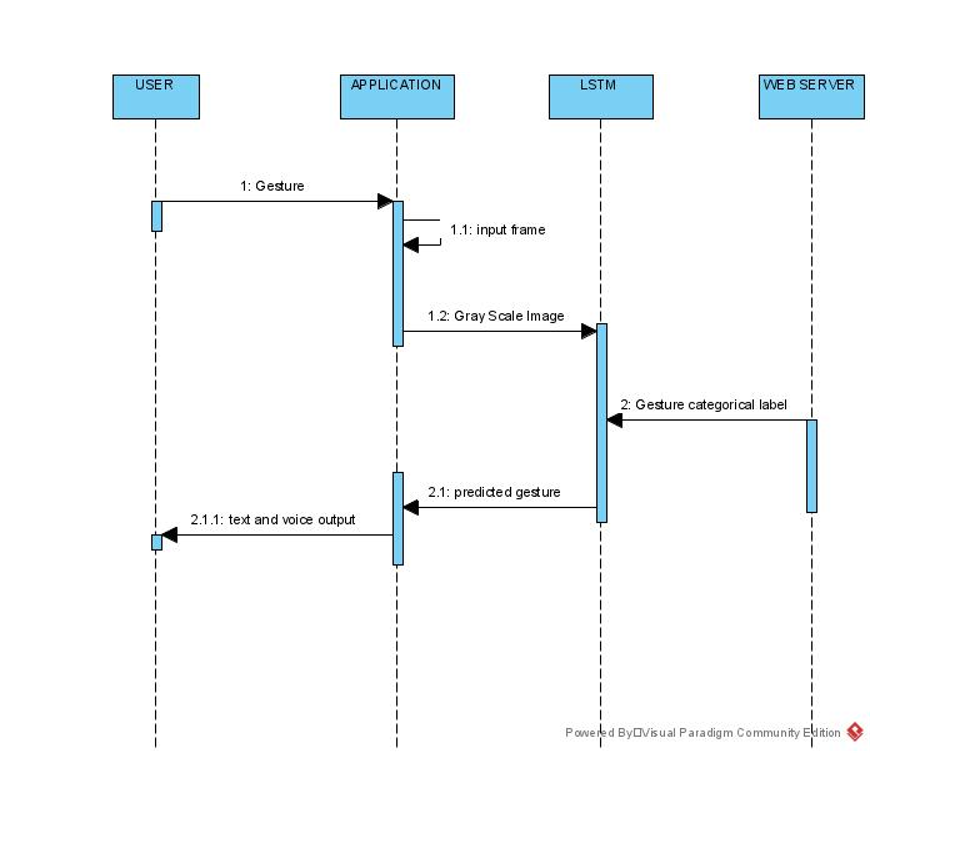
UC 7: Size

UC 8: Audio

UC 9: Scale

|  |  |  |
| --- | --- | --- |
| **Req’t** | **PW** | **UC1 UC 2 UC 3 UC 4 UC 5 UC 6 UC 7 UC 8 UC9** |
| **REQ 1** | **5** | **X** |
| **REQ 2** | **5** | **X X** |
| **REQ 3** | **4** | **X** |
| **REQ 4** | **1** | **X X** |
| **REQ 5** | **2** | **X** |
| **REQ 6** | **4** | **X X** |
| **REQ 7** | **1** | **X** |
| **REQ 8** | **2** | **X X** |
| **REQ 9** | **1** | **X** |
| **REQ 10** | **2** | **X** |
| **REQ 11** | **2** | **X** |
| **REQ 12** | **2** | **X X** |
| **REQ-13** | **2** | **X** |
| **MAX PW** | | 5 5 2 4 2 2 2 5 2 |
| **TOTAL PW** | | 11 9 3 10 2 3 2 5 2 |

SEQUENCE DIAGRAM:

****

LSTM = LONG SHORT-TERM MEMORY Machine learning model that we have used.

Model skeleton:

Training code skeleton:

! pip install opencv-python mediapipe sklearn matplotlib numpy tensorflow

import cv2

import numpy as np

import os

from matplotlib import pyplot as plt

import time

import mediapipe as mp

mp\_holistic = mp.solutions.holistic

mp\_drawing = mp.solutions.drawing\_utils

def mediapipe\_detection(image, model):

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

image.flags.writeable = False

results = model.process(image)

image.flags.writeable = True

image = cv2.cvtColor(image, cv2.COLOR\_RGB2BGR)

return image, results

def draw\_landmarks(image, results):

# mp\_drawing.draw\_landmarks(image, results.facemesh, mp\_holistic.FACEMESH) # Draw face connections

# Draw pose connections

mp\_drawing.draw\_landmarks(image, results.pose\_landmarks, mp\_holistic.POSE\_CONNECTIONS,

mp\_drawing.DrawingSpec(color= (80,22,10), thickness=2, circle\_radius=4),

mp\_drawing.DrawingSpec(color= (80,44,121), thickness=2, circle\_radius=2))

# Draw left hand connections

mp\_drawing.draw\_landmarks(image, results.left\_hand\_landmarks, mp\_holistic.HAND\_CONNECTIONS,

mp\_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle\_radius=4),

mp\_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle\_radius=2))

# Draw right hand connections

mp\_drawing.draw\_landmarks(image, results.right\_hand\_landmarks, mp\_holistic.HAND\_CONNECTIONS,

mp\_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle\_radius=4),

mp\_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle\_radius=2))

def extract\_keypoints(results):

pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.pose\_landmarks.landmark]). flatten () if results.pose\_landmarks else np.zeros(33\*4)

# face = np.array([[res.x, res.y, res.z] for res in results.face\_landmarks.landmark]). flatten () if results.face\_landmarks else np.zeros(468\*3)

lh = np.array([[res.x, res.y, res.z] for res in results.left\_hand\_landmarks.landmark]). flatten () if results.left\_hand\_landmarks else np.zeros(21\*3)

rh = np.array([[res.x, res.y, res.z] for res in results.right\_hand\_landmarks.landmark]). flatten () if results.right\_hand\_landmarks else np.zeros(21\*3)

return np.concatenate([pose, lh, rh])

result\_test = extract\_keypoints(results)

DATA\_PATH = os.path.join('MP\_Data')

# Actions that we try to detect

listOfWords = ['test']

actions = np.array(listOfWords)

# Thirty videos worth of data

no\_sequences = 30

# Videos are going to be 30 frames in length

sequence\_length = 30

for action in actions:

for sequence in range(no\_sequences):

try:

os.makedirs(os.path.join(DATA\_PATH, action, str(sequence)))

except:

Pass

cap = cv2.VideoCapture(0)

# Set mediapipe model

with mp\_holistic.Holistic(min\_detection\_confidence=0.5, min\_tracking\_confidence=0.5) as holistic:

# NEW LOOP

# Loop through actions

for action in actions:

# Loop through sequences aka videos

for sequence in range(no\_sequences):

# Loop through video length aka sequence length

for frame\_num in range(sequence\_length):

# Read feed

ret, frame = cap.read()

# Make detections

image, results = mediapipe\_detection(frame, holistic)

#Print(results)

# Draw landmarks

draw\_landmarks(image, results)

# NEW Apply wait logic

if frame\_num == 0:

cv2.putText(image, 'STARTING COLLECTION', (120,200),

cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,255, 0), 4, cv2.LINE\_AA)

cv2.putText(image, 'Collecting frames for {} Video Number {}'. format (action, sequence), (15,12),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE\_AA)

# Show to screen

cv2.imshow('OpenCV Feed', image)

cv2.waitKey(2000)

else:

cv2.putText(image, 'Collecting frames for {} Video Number {}'. format (action, sequence), (15,12),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE\_AA)

# Show to screen

cv2.imshow('OpenCV Feed', image)

# NEW Export keypoints

keypoints = extract\_keypoints(results)

npy\_path = os.path.join(DATA\_PATH, action, str(sequence), str(frame\_num))

np.save(npy\_path, keypoints)

# Break gracefully

if cv2.waitKey(10) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

GUI Mockups:

Predicting “K” letter correct

A picture containing person, person

Description automatically generated

Predicting “w” letter correct

A person holding a pair of glasses

Description automatically generated with low confidence

Predicting “L” letter correct on the web-app.

A picture containing text, person, screenshot

Description automatically generated

Implementation Plan

Phase I

• Generating Dataset

• Pre-processing Data

• Cleaning Data

• Data Analysis

Phase II

•Choosing Model

• Creating Model

• Training the model

• Validating the model

• Testing model on the dataset

• Checking accuracy of the model

Phase III ‘

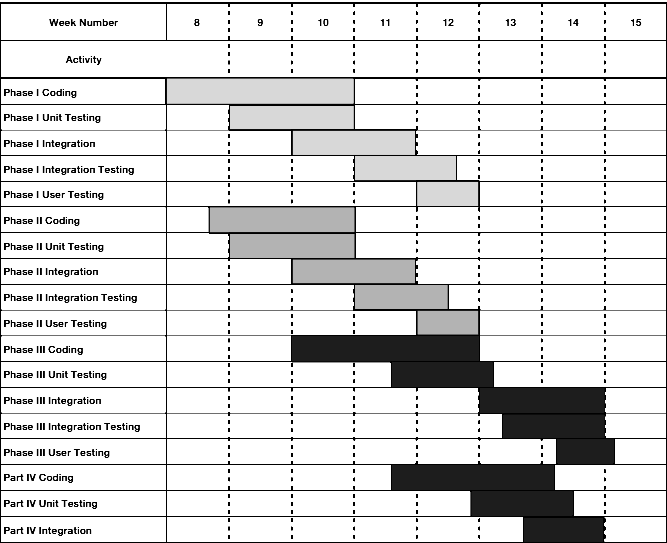
• Creating a local machine web application

• Integrating the model and the web application

• Creating a basic front end

• Testing the web application

Implementation Breakdown



Implementation Overview

The implementation of the application and model will be approached in a “Top-Down” fashion, where the Phase I would be accomplished first and then moving down further accordingly as shown in the implementation plan. Individual stubs will have to be written to test some methods and the correct performance of the main classes and how they are interacting with the overall end-product. Our implementation will be broken up into 3 phases.

The first phase will consist of the mainly generating the dataset and then preprocessing it, creating a proper dataset will be the driving force behind the software.

Phase two will consist of the creation of the novel Machine learning model that will be trained and tested on the dataset that is being created in the phase I. This model will help us in predicting the gestures that a user is doing via the device camera and the predicted gesture alphabets would be shown above on the device screen.

Phase three will contain the remaining portions of the use cases. Unit testing will be performed along the way a simple web user-interface would be created that would help us to use this model via a web-app coded in Flask.

All coding will be done through Jupyter Notebook and Vs code on the Windows platform and the coding will be done in Python and Flask framework. Testing will be done on a Windows Laptop to get real time results and to aid in visual and program debugging.

**Test Cases**

**Image**

**Test -case Identifier**: TC-1

**Use Case Tested:** UC-1, main success scenario

**Pass criteria**: The test passes if the user shows the sign that is contained in the database.

**Input Data:** images

|  |  |
| --- | --- |
| Test procedure: | Expected Result: |
| Step 1: If we show the wrong gesture, which is not in database. | It records unsuccessful attempt and prompts user to try again. |
| Step 2: Show the images with gesture. | System takes the data and convert into text. |
|  |  |

**Microphone**

**Test -case Identifier:** TC-2

**Use Case Tested**: UC-2, main success scenario

**Pass criteria**: The test passes if the microphone is being used by the program.

**Input Data**: voice

|  |  |
| --- | --- |
| Test procedure: | Expected Result: |
| Step 1: If we give the audio which is not in database. | Program shows to record the voice again. |
| Step 2: We have to start recording voice | Program starts converting the audio to text and gestures. |

**Text**

**Test -case Identifier:** TC-3

**Use Case Tested**: UC-3, main success scenario

**Pass criteria**: The test passes if the text is converted gestures on screen.

**Input Data:** text

|  |  |
| --- | --- |
| Test procedure: | Expected Result: |
| Step 1: If we give the text which is not in database. | It will show the textbox again to give another text. |
| Step 2: We have to start giving text. | Program starts converting the text into gestures. |

**Gesture**

**Test -case Identifier:** TC-4

**Use Case Tested:** UC-4, main success scenario

**Pass criteria:** The test passes if it converts gestures to speech.

**Input Data:** gesture

|  |  |
| --- | --- |
| Test procedure: | Expected Result: |
| Step 1: If we give the gesture which is not in database. | It will show the prompt again to show gesture again. |
| Step 2: We have to show gesture. | Program starts converting the gesture into speech. |

**Speaker**

**Test -case Identifier:** TC-5

**Use Case Tested**: UC-5

**Pass criteria:** The test passes if user given audio is output via speakers.

**Input Data:** gesture

|  |  |
| --- | --- |
| Test procedure: | Expected Result: |
| Step 1: If we give the audio which is not in database. | It will show the prompt again to give audio again. |
| Step 2: We have to give gesture. | Program allow access and gives the audio through speakers. |

**Navigate**

**Test -case Identifier:** TC-6

**Use Case Tested:** UC-6

**Pass criteria**: The test passes if user navigates through all options

**Input Data:** navigate

|  |  |
| --- | --- |
| Test procedure: | Expected Result: |
| Step 1: If we try navigate and are not able to do it properly through all options available | It will show the prompt again to select the desired option |
| Step 2: We have to select one mode. | Program lets us to use the desired option. |

**Size**

**Test -case Identifier**: TC-7

**Use Case Tested:** UC-7, main success scenario

**Pass criteria**: The test passes if user performs sign language gestures.

**Input Data:** size

|  |  |
| --- | --- |
| Test procedure: | Expected Result: |
| Step 1: If we give the size of the samples that which is not in database | It will show the prompt again to give size of the training sample again. |
| Step 2: We have to capture the samples | Program gives the N number of images to be used for training |

**Audio**

**Test -case Identifier:** TC-8

**Use Case Tested:** UC-8, main success scenario

**Pass criteria:** The test passes if we give the size of training sample.

**Input Data:** audio

|  |  |
| --- | --- |
| Test procedure: | Expected Result: |
| Step 1: If we give the audio that is not in database. | It will show the prompt again to show the gesture |
| Step 2: We have to perform sign language gesture | Output will be the speech which converted from text obtained from model |

# 

# Test coverage

The degree to which a software program's specification or code has been exercised by tests is measured by test coverage.

The percentage of a program's source code that has been tested is called code coverage.

**CODE COVERAGE:**

Equivalence testing is a black box testing method that partitions the space of all potential inputs into equivalence groups so that the program performs the same on each group.

Equivalence classes:

|  |  |  |
| --- | --- | --- |
| **Test Scenario #** | **Test Scenario Description** | **Expected Outcome** |
| 1 | Show one hand gesture which is not in database | System should not accept |
| 2 | Show one gesture which is in data | System will convert it into text |
| 3 | Show one gesture which is related to database | System will convert it into text |

**State based testing example:**

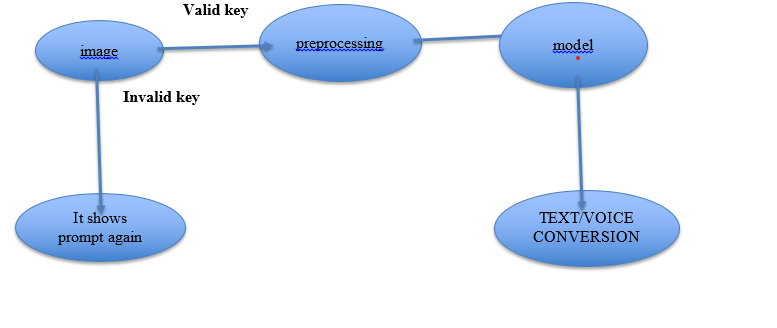
State-based testing establishes a set of abstract states that a software unit can take and compares the unit's actual states to the expected states to test its behavior.

Object-oriented systems have taken to this technique.

An object's state is defined as a restriction on the values of its attributes.

Because the attributes are used to compute the object's behavior, the behavior is determined by the object's state.

**Example:**



**Practical Aspects of Unit Testing**

1. Mock objects:
   1. An image simulates the part of the system that invokes text on the gesture component
   2. An image simulates the gesture into text
2. The unit to be tested is also known as the **gesture**
3. Unit testing follows this cycle:
   1. Create the thing to be tested (gesture)
   2. After preprocessing is done it will be shown as a text
   3. Evaluate the text that is same as shown in the gesture
   4. Text and voice both will be expected output  
        
         
        
         
        
         
        
         
        
         
        
         
        
         
        
         
        
         
        
         
         
        
      **Problem Frames**

**Problem frame for Use Case 1 (Camera):** To use the camera on which the program is running

**Requirements involved in Use Case 1:** Req 1, Req 5, Req 6

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Requirement** | **Problem Frame** |
| Req 1 | The program should be able to use the device’s camera to take the key points from the data processed to it. | Commanded behavior |
| Req 5 | The program should be asking the user whether he wants to allow the software to use the device camera or not | Simple Workpieces |
| Req 6 | The program should be predicting the gestures accurately. | Required behavior |

**Problem frame for Use Case 4 (Gesture):** To be able to show the gestures to the camera so that it reads the user’s hand correctly

**Requirements involved in Use Case 2:** Req 3, Req 4, Req 6, Req, 7

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Requirement** | **Problem Frame** |
| Req 3 | The program must have a user-interface, where the user can use the device camera for showing gestures so that the model takes that data and converts it into text. | Simple workpieces |
| Req 4 | The program’s GUI should also have a textbox, where the user can type in the text which could be converted to gesture language. | Information display |
| Req 6 | The program should be predicting the gestures accurately | Required behavior |
| Req 7 | The program should be able to recognize the hand of the user correctly | Transformation |

**Required behavior:**

**Use Case 1: Camera Access – REQ 6: predicting the gestures accurately**

**Existing behavior for REQ 6:**

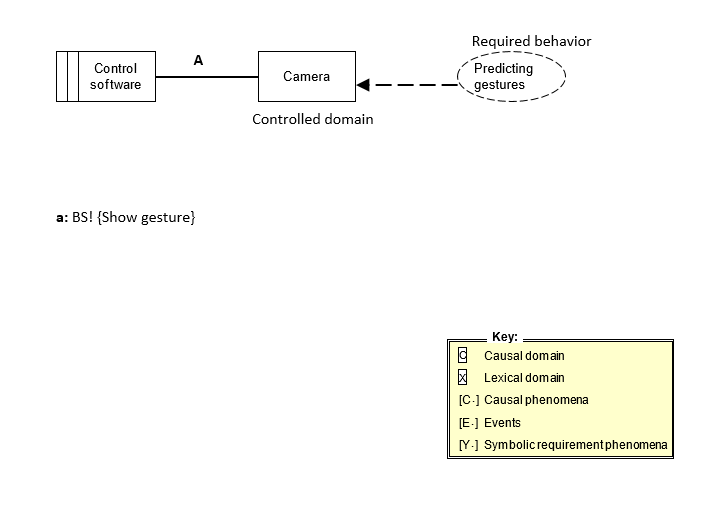
Currently, the gestures given by the user are read correctly until the user gives some unknown gesture. The model can’t read it properly and gives wrong results

**Required behavior for REQ 6:**

The model would be able to recognize it is an unknown gesture and tell the user that this gesture is not in the database.

**Capabilities available:**

The bone structure of the hand is being read accurately



**Required behavior:**

**Use Case 4: Gesture – REQ 6: predicting the gestures accurately**

**Existing behavior for REQ 6:**

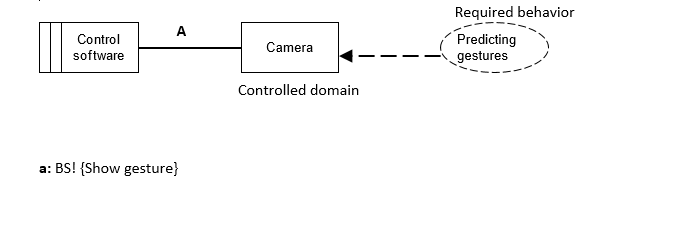
Currently, the gestures given by the user are read correctly until the user gives some unknown gesture. The model can’t read it properly and gives wrong results

**Required behavior for REQ 6:**

The model would be able to recognize it is an unknown gesture and tell the user that this gesture is not in the database.

**Capabilities available:**

The bone structure of the hand is being read accurately



**Commanded behavior:**

**Use Case 1: Camera Access – REQ 1: accessing the camera to take the keypoints**

**Requested user commands (REQ 1):**

Use a camera and show hand gestures

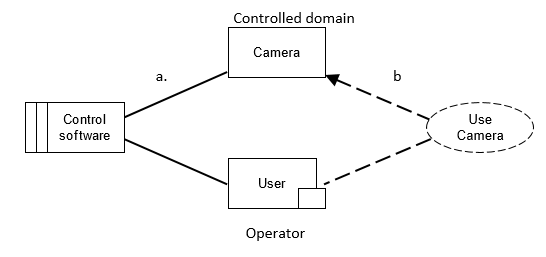
**Command applicability under different scenarios REQ 1:**

Under training data, the gesture prescribed can be shown.

Gesture already trained can be predicted easily

**Consequences of unacceptable commands REQ 1:**

The model is not able to predict the gesture correctly and no output is shown.



a: TS! {Open

b: TR! {Use camera[i]} [Y2]

**Information Display:**

**Use Case 4: Gesture – REQ 4: have a textbox to show the text.**

**Information to observe for REQ 4:**

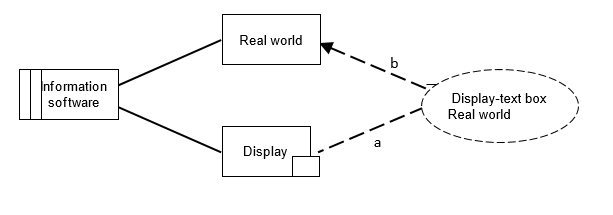
The text converted gestures existing in the database

**Required information to visualize for REQ 4:**

Access information through text will be displayed as store without post-processing

**Rules for visualization for REQ 4:**

Render the result of the text query as an HTML table.



a: TS! {Display text[i]} [E1]

b: TR! {Show-text-to-real world[i]}

**Simple Workpieces:**

**Use Case 1: Camera – REQ 5: ask user to allow access to camera**

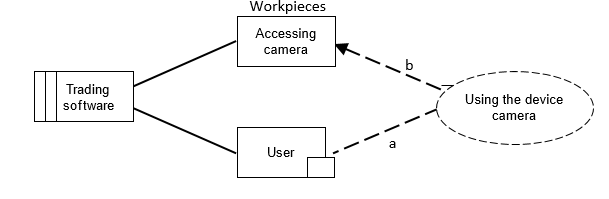
**Requested commands for REQ 5:**

Allow camera access

**Command applicability under different scenarios REQ 5:**

User denied access

User wants to change the permission of camera access



a: TS! {Camera access} [E1]

b: TR! {Using the camera[i]} [Y2]

**Simple Workpieces:**

**Use Case 4: Gesture – REQ 3: having a user-interface**

**Data structures for REQ 3:**

Database storing the gestures collected as arrays from the user-interface to the backend

**Requested commands for REQ 3:**

Show gesture.

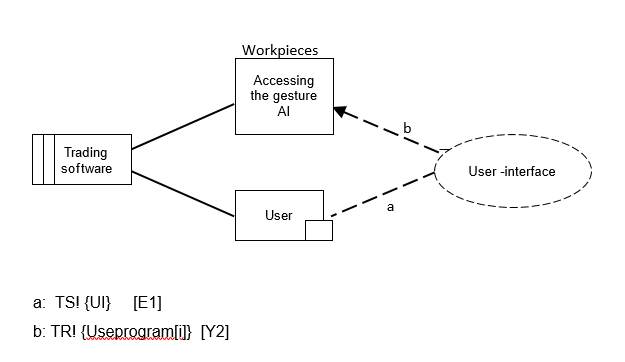
Type text.

Use training data.

**Command applicability under different scenarios REQ 3:**

Gesture not in the database.

Gesture prediction was not accurate.



**Transformation:**

**Use Case 4: Gesture – REQ 7: recognizing the user hand bone structure correctly**

**Data structures for REQ 7:**

Database storing the gestures collected as arrays.

**Requested commands for REQ 7:**

Show gesture.

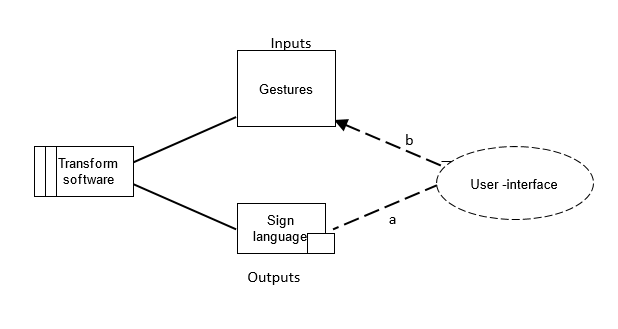
Type text.

Use training data.

**Command applicability under different scenarios:**

Gesture not in the database.

Gesture prediction was not accurate

a: TS! {UI} [E1]

b: TR! {Useprogram[i]} [Y2]

Source Code:

Code in file uploaded with the project assignment named GestureAI.ipynb