CONVERSION OF PAKISTAN SIGN LANGUAGE INTO TEXT AND SPEECH USING MACHINE LEARNING

Submitted to:

Department of Computer Sciences & IT

The University of Lahore

Submitted by:

Muhammad Saad Qadri
BCS02153218
Saadbutt32@gmail.com

Muhammad Junaid Ejaz
BCS02153244
shapedprodigy@gmail.com

Supervised by:

Dr. Mudasser Naseer
Associate Professor
Department of Computer Sciences & IT

SIGNATURE PAGE

Muhammad Junaid Ejaz	BCS02153244	
Muhammad Saad Qadri	BCS02153218	

Approved by:

_____ Date of Submission

_____ Advisor

Dr. Mudasser Naseer

ABSTRACT

Communication is a primary human need and language is the medium for this. Most people have the ability to listen and speak and they use different languages to communicate. Hearing impaired people use signs to communicate with other people. Pakistan Sign Language (PSL) is the preferred language of the deaf people in Pakistan. In order to make the communication between deaf and normal people simpler and easier, computer based PSL interpreters are required.

The aim of the project is to bridge the communication gap between signers and non-signers by developing an interpretation system which is cost effective and easy to use. The system is expected to have the ability to convert different PSL signs into text and speech. It will also allow users to learn PSL interactively.

The system works by taking images of user making a particular PSL sign through webcam and uses skeletal tracking combined with machine learning to detect what sign they are making. Currently there is no publicly available dataset for PSL so, we have introduced a new PSL dataset consisting of 37 Urdu alphabets and 12 Urdu words. The dataset contains human body key points that are estimated by OpenPose and each data point in the dataset is annotated with an Urdu label.

Different machine learning models were considered but we found Artificial Neural Network (ANN) to be the most suitable for this task. We were surprised to find that the optimal ANN design came from a simple architecture, rather than a complicated one. The system recognizes almost all the signs with good accuracy in real time. We have trained two ANN models for alphabet and word recognition. The alphabet recognition model has achieved 99.4% accuracy on test data and the word recognition model has achieved 99.2% accuracy on test data.

DEDICATION

We dedicate this project to ALLAH Almighty our creator, our strong pillar, our source of inspiration, wisdom, knowledge and understanding. He has been the source of our strength throughout this program.

ACKNOWLEDGEMENT

The countless blessings of ALLAH Almighty have made this project easy for us. We present our humble gratitude for Him, for the shower of His special blessings on us. We take this opportunity to acknowledge the role of those whose endless efforts and contributions were always there when we needed them. We would like to thank our parents for their support that made us confident in life and to take challenges and pursue them. We would also like to thank our supervisor Dr. Mudasser Naseer for his precious time and considerate guidance

TABLE OF CONTENTS

List Of Figures

LIST OF TABLES

CHAPTER 1: INTRODUCTION TO THE PROBLEM

1.1. INTRODUCTION

Communication is a primary human need and language is the medium for this. Most people have the ability to listen and speak and they use different languages like Urdu, English, Swedish etc. to communicate with each other. Hearing impaired people use signs to communicate. Pakistan Sign Language (PSL) is the preferred language of the deaf people in Pakistan. However, a person that do not know sign language face difficulties in communicating with deaf people. PSL interpreters are required to facilitate communication between the deaf and hearing. Without the help of an interpreter the communication among the deaf and other people may be impaired or nonexistent. In this situation, a system that can convert PSL into text and speech will be highly helpful.

In this project we try to develop a system to convert PSL into Urdu text and speech for communication between signers and non-signers. Main objective is to facilitate a large population of hearing-impaired persons and making them an integral part of the society. The system is divided into 3 main modules, detection module, learning module and capture module. The detection module has two modes one for alphabet detection and other for word detection. The learning module is used for interactive PSL learning and the capture module is used for adding new data to dataset.

The detection system works by taking images of user making particular sign through webcam and uses skeletal tracking combined with machine learning to detect what sign they are making.

The automated dataset capturing system works by capturing images of user for specified time, extracting the key points of images using OpenPose, deleting the images in which OpenPose shows less confidence, plotting the skeleton of remaining images and showing them to the user, the user then selects the plotted images that does not match the sign he was making and deletes them. The user is then asked to enter the label for remaining images and these images are stored by the system in a folder named as the value of label entered by user.

The learning system works by showing the user images of different PSL signs and prompting them to make that sign, the system then checks whether they have made the correct sign or not and displays a message accordingly.

1.2. PURPOSE

A noticeable amount of deaf community exists in Pakistan. In a detailed study based upon statistical data regarding the deaf people in Pakistan [1][2], It is stated that about 3.3 million Pakistanis are suffering from different kinds of disability in which 0.24 million are impaired of hearing which approximates to 7.4% of overall disables. A very important point is that 55% of the total disabled lie in the age group from 5 -29 years.

Whenever deaf people need to communicate with unimpaired people in any situation, they either need to hire a translator or request the unimpaired people to communicate through writing. The existing translators are very difficult to get all the time and they are very expensive as well.

The interaction of deaf with normal people becomes very slow and embarrassing. In this regard, deaf people find it difficult to do what they want to do at shopping malls, banks and post offices etc. Moreover, deaf people in Pakistan mostly end up having friends that are also deaf. There is definitely a need for an automated system that can facilitate communication between deaf and unimpaired people.

There exist many sign language interpreters that are in English or follow American Sign Language but there are not many interpreters who follow PSL.

1.3. OBJECTIVE

The aim of the project is to bridge the communication gap between signers and non-signers by developing an interpretation system which is cost effective and easy to use. The system is expected to have the ability to convert different PSL signs into text and speech.

The main objectives of this system are:

- Make the sign language interpretation cheap by introducing an automated system that does not require any specialized sensors.
- Provide an interactive learning system for PSL.
- Make the system more accurate and easier to use.

1.4. EXISTING SOLUTION

Research in the sign language system has two well-known approaches, image processing and data Glove. Worldwide efforts have been made to aid the deaf community in communicating with non-signers. Some of the known solutions are:

- A research-based development of a sign language recognition system using a data glove is proposed by Nicholas Born [4]. In this approach, detection of hand is done by the sensor glove that consists of flex sensor and accelerometer.
- Most recently Muhammad Wasim et al [5] used leap motion sensor to develop a system to communicate in sign language. They were able to detect some signs with an accuracy level of up to 100%.
- Another approach used by A. Muhammad at el [3] is to track skeleton using Kinect sensor. The Accuracy of system is not known.
- Another approach using data glove but this time the colored one, is proposed by Sumaira et al [6]. They have used a fuzzy classifier to recognize the signs/gestures performed by the deaf. Their algorithm uses the angle between fingertip and joint for classification of gesture. Their dataset was based on Urdu alphabets of Pakistan sign language. They have achieved an overall accuracy of 95% as 35 out of 37 alphabets were recognized correctly. Again, the cost of color data glove, recognition of single hand gestures and static gestures are the limitations of their system.
- Another work for reducing communication gap between the deaf and hearing is done by Asif Ali [7]. He has proposed a system which takes input in both forms text and image of sign and convert it into other forms. They have performed this for Urdu alphabets of PSL

using Haar classifier. They have used a simple RGB camera for this purpose. However, they have not specifically mentioned the size and nature of the dataset used for experimentation. The accuracy rate of the developed system is also missing.

The main drawback of most of the above-mentioned systems is the use of special equipment such as gloves, leap motion sensor or depth sensor (i.e. Microsoft Kinect), which are expensive and usually not available in real life scenarios. Another problem is that these sensors need a closed environment such as the Kinect sensor cannot work properly in sunlight.

1.5. Proposed Solution (Gap Analysis)

We propose a PSL recognition system based on the human skeletal key points that are estimated by third party library OpenPose. The OpenPose is an open source toolkit for real-time multi-person key point detection it can estimate in total 130 key points where 18 key points are from body, 21 key points are from each hand, and 70 key points from a face.

Currently there is no publicly available dataset for PSL, especially for use with OpenPose. In case of sign language dataset, it is more difficult to collect than the other dataset because, for the accuracy of the data, professional signers should be used for recording sign language videos of high quality, which are hard to find. So, we have decided to develop our own dataset for PSL words and alphabets. Different machine learning algorithms are used for recognizing signs and are analyzed for accuracy and performance. Machine Learning libraries like scikit-learn, TensorFlow and keras are also used.

Our recognition system is robust in different backgrounds as it only detects the human body. The system based on the human key point detection works well regardless of signer since the variance of extracted key points is negligible. Moreover, we normalize the feature set using the mean and standard deviation. Lastly, the use of high-level features is necessary when the scale of the dataset is not large enough.

CHAPTER 2: SOFTWARE REQUIREMENT SPECIFICATION

2.1. INTRODUCTION

2.1.1. PURPOSE

The purpose of this document is to present a detailed description of Pakistan Sign Language conversion system. It will explain the purpose and features of the system, the interfaces of the system, what the system will do and the constraints under which it must operate. This document is intended for both the stakeholders and the developers of the system.

2.1.2. SCOPE

This software system is a desktop application for converting Pakistan Sign Language into text and speech. This system will be used by hearing impaired people to communicate with normal hearing people, without any specialized equipment. By minimizing the cost and availability issues and maximizing the output, the system meets users need while remaining easy to understand and use.

2.1.3. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

TABLE 2.1 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

Term/Abbreviation	Description
OPENPOSE	This is a convolutional neural network key point detector of human body. [8]
PYTHON	Python is an interpreter, high-level, general-purpose programming language.
TensorFlow	TensorFlow [13] is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms.
Eel	Eel is a software library that allows the development of simple desktop GUI like applications using front-end components of web applications and back-end in Python/JavaScript.

SVM	Support Vector Machine SVM is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.[9]
KNN	K-Nearest Neighbors KNN is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).[10][11]
ANN	Artificial Neural Networks ANN or Neural Networks are computational algorithms. It intended to simulate the behavior of biological systems composed of “neurons”. ANNs are computational models inspired by an animal's central nervous systems. It is capable of machine learning as well as pattern recognition.[12]
PSL	Pakistan Sign Language.
RNN	Recurrent Neural Network RNN is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.
SD	Standard Deviation SD is a quantity expressing by how much the members of a group differ from the mean value for the group.
JSON	JavaScript Object Notation JSON is a lightweight format for storing and transporting data.

2.1.4. OVERVIEW

The second section, the Overall Description section of this document, gives an overview of the functionality of the product and describes the informal requirements of the product.

The third section, Requirements Specification section of this document describes the functionalities of the product in technical terms.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences.

2.2. OVERALL DESCRIPTION

2.2.1. PRODUCT PERSPECTIVE

The proposed application is based on the human skeletal key points that are estimated by third party library OpenPose. The OpenPose is an open source toolkit for real-time multi-person key point detection it can estimate in total 130 key points. The system takes images of user making particular sign through webcam and uses skeletal tracking combined with machine learning to detect what sign they are making.

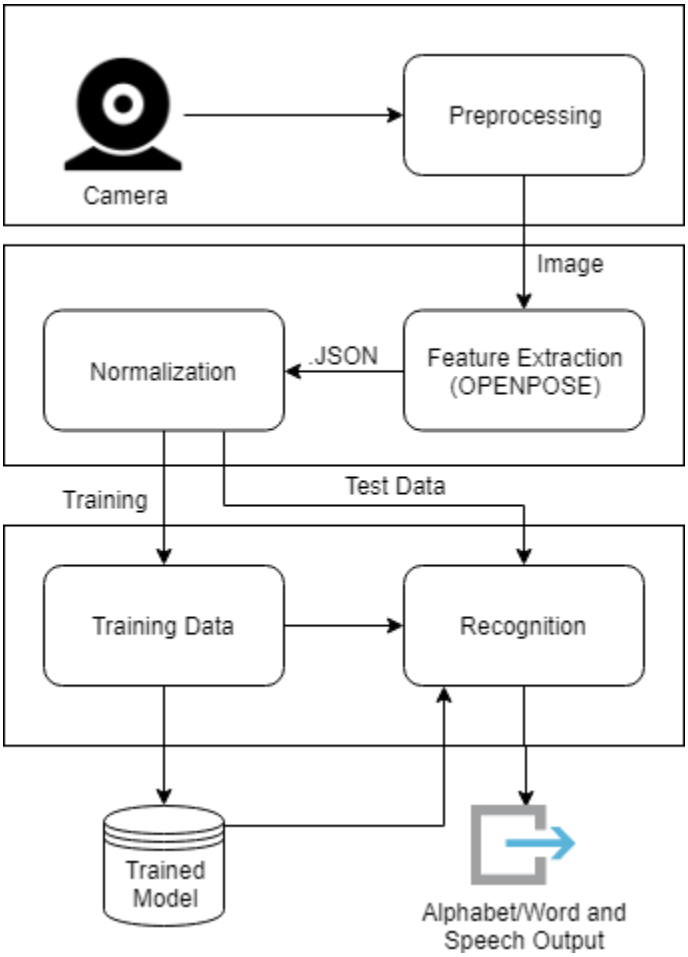


Figure 2.1. Product Perspective

USER INTERFACES

This project can serve both speech-disordered people and the others. Yet it is designed especially for the speech-disordered people's convenient use.

The system is divided into 3 main modules, detection module, learning module and capture module. The detection module has two modes one for alphabet detection and other for word detection. The learning module is used for interactive PSL learning and the capture module is used for adding new data to dataset.

MAIN SCREEN

Figure 2.2. Main Screen

Detection Screen

Figure 2.3. Detection screen

Capture Screen

Figure 2.4. Capture Screen

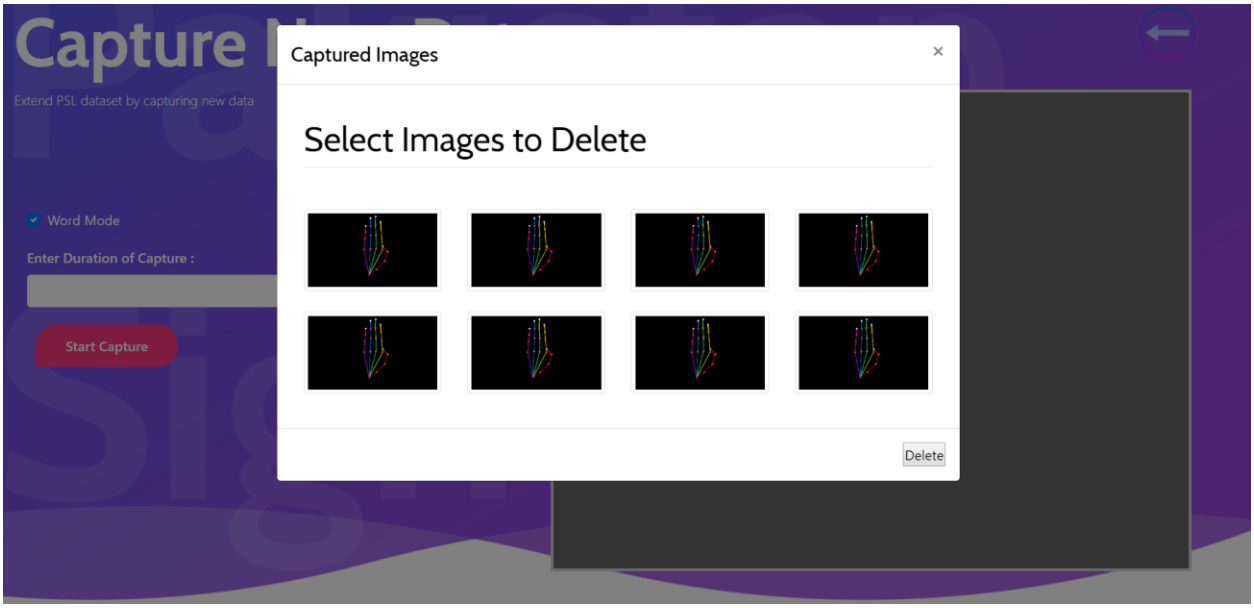


Figure 2.5. Delete Image

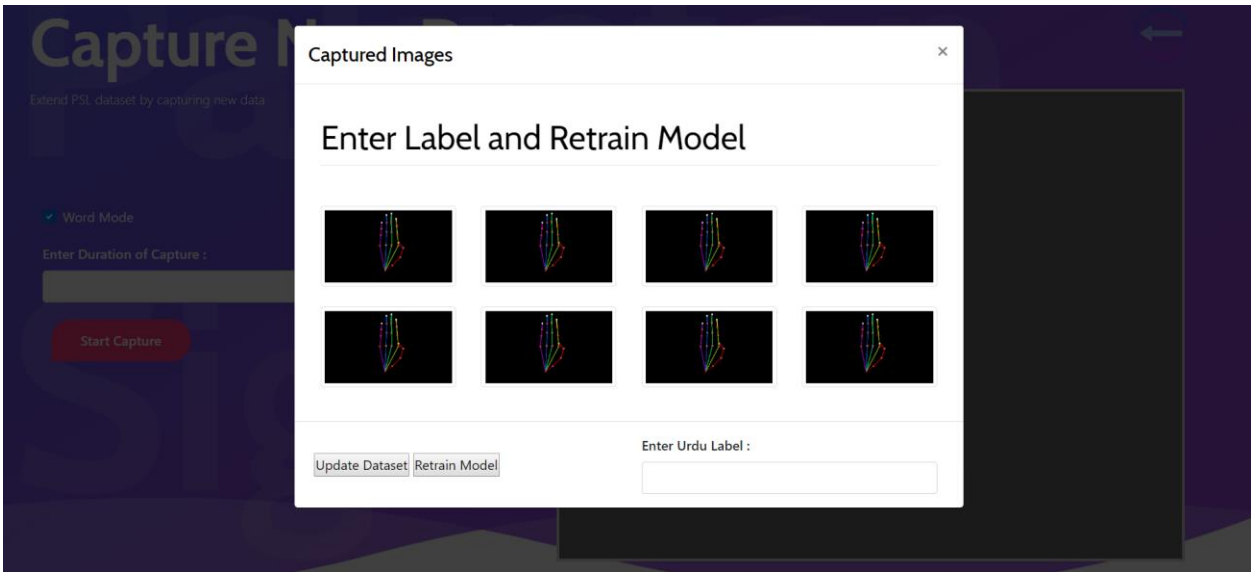


Figure 2.6. Retrain model

Hardware interfaces

Following hardware components are required for system to work properly:

- Cuda compatible Nvidia graphic card.
- Working webcam.

Software interfaces

The software will operate in Windows 10 environment, with cuda compatible NVidia drivers installed.

Communications Interfaces

There are no communications interfaces for the system.

Memory

At least 4 GB of Ram is needed for the system.

2.2.2. PRODUCT FUNCTIONS

The main purpose of our application is to aid communication between signer and non-signer. The user will perform a sign in front of webcam, sign will be recognized by the system and system will output it in the form of text and speech.

START DETECTION

TABLE 2.2 START DETECTION

ID:	FR_01			
Name:	Start Detection			
Description	Input	Output	Requirements	Basic Workflow
Start recognizing signs	Click Start Detection button	Recognition Started	Recognition Stopped	Click Start Detection button to start recognizing signs

RECOGNIZE SIGN

TABLE 2.3 RECOGNIZE SIGN

ID:	FR_02			
Name:	Stop Detection			
Description	Input	Output	Requirements	Basic Workflow
Recognize specific sign made by user and convert it into text and speech	Valid sign in front of the webcam	Correct conversion into text and speech	Recognition has started, Valid sign made by user in front of webcam.	User Signs in front of camera, the system recognizes the sign and convert it into text and speech.

STOP DETECTION

TABLE 2.4 STOP DETECTION

ID:	FR_03			
Name:	Stop Detection			
Description	Input	Output	Requirements	Basic Workflow
Stop recognizing signs	Click Stop Detection button	Recognition Stopped	Recognition Started	Click Stop Detection button to stop recognizing signs

ENABLE SPEECH

TABLE 2.5 ENABLE SPEECH

ID:	FR_04			
Name:	Enable Speech			
Description	Input	Output	Requirements	Basic Workflow
Enable speech output	check the ‘enable speech’ checkbox	speech output enabled	‘enable speech’ checkbox unchecked	check the ‘enable speech’ checkbox to Enable speech output

DISABLE SPEECH

TABLE 2.6 DISABLE SPEECH

ID:	FR_05			
Name:	Disable Speech			
Description	Input	Output	Requirements	Basic Workflow
Disable speech output	uncheck the enable speech checkbox	speech output disabled	enable speech checkbox checked	uncheck the enable speech checkbox to disable speech output

Display Recognition Output

Table 2.7 Recognition Output

ID:	FR_06			
Name:	Display Recognition Output			
Description	Input	Output	Requirements	Basic Workflow
display recognized sign output on screen	Recognition Output	Output Displayed	Detection started; some output received from recognition module	System gets input from recognition module and displays it on screen

Play Speech Output

Table 2.8 Speech Output

ID:	FR_07			
Name:	Play Speech Output			
Description	Input	Output	Requirements	Basic Workflow
play speech of recognized sign	Recognition Output	speech played	speech enabled; some output received from recognition module	System gets input from recognition module and play the speech

Switch to word mode

Table 2.9 Switch to word mode

ID:	FR_08			
Name:	Switch to Word Mode			
Description	Input	Output	Requirements	Basic Workflow
Switch to word mode from alphabet mode	check the ‘switch to word mode’ checkbox	Switched to word mode from alphabet mode	‘switch to word mode’ checkbox unchecked	check the ‘switch to word mode’ checkbox to Switch to word mode from alphabet mode

Switch to Alphabet Mode

Table 2.10 Switch to Alphabet mode

ID:	FR_09			
Name:	Switch to Alphabet Mode			
Description	Input	Output	Requirements	Basic Workflow
Switch to alphabet mode from word mode	uncheck the ‘switch to word mode’ checkbox	Switched to alphabet mode from word mode	‘switch to word mode’ checkbox checked	uncheck the ‘switch to word mode’ checkbox to Switch to alphabet mode from word mode

VIEW HELP

TABLE 2.11 VIEW HELP

ID:	FR_10			
Name:	View Help			
Description	Input	Output	Requirements	Basic Workflow
View Help Document	Click Help Button	Help document shown in new page	no requirements	user clicks help button, and system shows help document

DISPLAY SIGN IMAGE

TABLE 2.12 DISPLAY SIGN IMAGE

ID:	FR_11			
Name:	Display Sign Image			
Description	Input	Output	Requirements	Basic Workflow
display image of specific sign on screen and prompt user to try making that sign	Learning Module	Image displayed successfully	no requirements	user opens learning module, and system displays image of specific sign on screen

VALIDATE SIGN

TABLE 2.13 VALIDATE SIGN

ID:	FR_12			
Name:	Validate Sign			
Description	Input	Output	Requirements	Basic Workflow
Validate sign made by user against an image shown to him	webcam	Display success message	Valid sign made by user in front of webcam.	User Signs in front of camera, the system validates the sign and displays a success message

CHANGE SIGN IMAGE

TABLE 2.14 CHANGE SIGN IMAGE

ID:	FR_13			
Name:	Change Sign Image			
Description	Input	Output	Requirements	Basic Workflow
Show next sign image to user if previous sign was validated	Validation function	Image Changed successfully	previous sign was validated	The system shows next sign image to user if previous sign was validated

Skip Sign

Table 2.15 Skip Sign

ID:	FR_14			
Name:	Skip Sign			
Description	Input	Output	Requirements	Basic Workflow
Skip to next sign if user clicks skip button	Click ‘Skip’ Button	Image Changed successfully, moved to next sign	sign has not been validated	The system shows next sign image to user if user clicks ‘skip’ button

Select Capture Mode

Table 2.16 Select Capture Mode

ID:	FR_15			
Name:	Select Capture Mode			
Description	Input	Output	Requirements	Basic Workflow
Select either word mode or alphabet mode from ‘select mode’ dropdown	‘select mode’ dropdown	mode changed to selected dropdown value	Capture has not been started	select mode from the ‘select mode’ dropdown to select word mode or alphabet mode

Set Capture Time

Table 2.17 Set Capture Time

ID:	FR_16			
Name:	Set Capture Time			
Description	Input	Output	Requirements	Basic Workflow
Enter capture time in seconds, in ‘enter time’ text field to set time for capture process	‘enter time’ text field	time set successfully	Capture has not been started	User enters capture time in ‘enter time’ text field and system set time for capture process

Display Captured Images

Table 2.18 Display Captured Images

ID:	FR_17			
Name:	Display Captured Images			
Description	Input	Output	Requirements	Basic Workflow
Display images captured in capture process	images captured in capture process	images displayed on screen	some images received from capture process	the system displays images captured in capture process

Delete Invalid Images

Table 2.19 Delete Invalid Images

ID:	FR_18			
Name:	Delete Invalid Images			
Description	Input	Output	Requirements	Basic Workflow
Select images of invalid signs and delete them	selected images	images deleted successfully	some images selected to delete	user selects the images to be deleted and the system deletes them

Set Label Of Sign

Table 2.20 Set Label Of Sign

ID:	FR_19			
Name:	Set Label of Sign			
Description	Input	Output	Requirements	Basic Workflow
enter label in ‘set label’ text field corresponding to captured sign and save captured files with that label	‘set label’ text field	captured files saved successfully with entered label	some images captured in capture process	user enters label in ‘set label’ text field and the system save captured files with that label

Populate Dataset

Table 2.21 Populate Dataset

ID:	FR_20			
Name:	Populate Dataset			
Description	Input	Output	Requirements	Basic Workflow
Click ‘populate dataset’ button to update dataset with new captured files	Click ‘populate dataset’ button	dataset updated with new captured files	some images captured in capture process	user clicks ‘populate dataset’ button and the system updates dataset with new captured files

Retrain Model

Table 2.22 Retrain Model

ID:	FR_21			
Name:	Retrain Model			
Description	Input	Output	Requirements	Basic Workflow
Click ‘Train Model’ button to train ANN model with updated dataset	Click ‘Train Model’ button	ANN model trained with updated dataset	no requirements	user clicks ‘Retrain Model’ button and the system trains ANN model with updated dataset

2.2.3. USER CHARACTERISTICS

TABLE 2.23 USER CHARACTERISTICS

Name	Description
Real-Time User	A person who uses the system to convert PSL into text and speech in real time.
Learner User	A person who wants to learn PSL.

2.2.4. CONSTRAINTS

The current constraints on the project are related to hardware. At present, a working webcam, Core i3, 4 GB RAM and Cuda compatible Nvidia graphics card is required.

2.2.5. ASSUMPTIONS AND DEPENDENCIES

A number of factors that may affect the requirements specified in the SRS include:

- If the person does not perform the correct gesture of required alphabet/word the system will not produce the desired output.
- The hand of the user should be clearly visible to the camera to maximize efficiency.

2.2.6. APPORTIONING OF REQUIREMENTS

- The system will start detecting dynamic signs which are mostly used in words or sentences.
- Android and iOS apps in future versions.
- The dataset to detect more words will be added gradually.

2.3. Specific requirements

2.3.1. Functional Requirement

Sign Language Recognition:

The application must recognize signs of the user using the camera's inputs.

Literal translation:

The application must show on the screen the word or the idea that refers to the user movement detected.

Interactive learning:

The application must provide an interface for interactive learning of PSL.

2.3.2. Non-functional Requirements

Usability

- There will be two user types – the normal user and the hearing-impaired user – each of which will have its own corresponding interface.
- The minimal requirement what a normal user can do is learn gestures or type a word/alphabet which can be converted into gesture.
- The minimal requirement what a hearing-impaired user can do is perform and learn gestures.

Reliability

- The system must function under any given circumstances.
- There can be a maximum of 1 bug/KLOC.
- In the case of bad gesture recognition, the system will give an appropriate error.

Performance

- The recognition process will be done in real-time. (3 seconds maximum)
- The maximum response time to get the gesture animation will be 3 seconds.

Design Constraints

- The design of the system will be made to cater to both normal and people with hearing impaired.
- Most of the coding will be done in Python.

Portability

The system will be easily available and should be portable for both users.

Maintainability

- The system will be regularly tested for any types of glitches and errors which can affect its efficiency.
- It will be regularly updated to prevent any virus.

License Agreement

It will be available under a GPL license and will be open-source.

CHAPTER 3: USE CASE ANALYSIS

3.1. USE CASE DIAGRAM-(LEVEL-1)

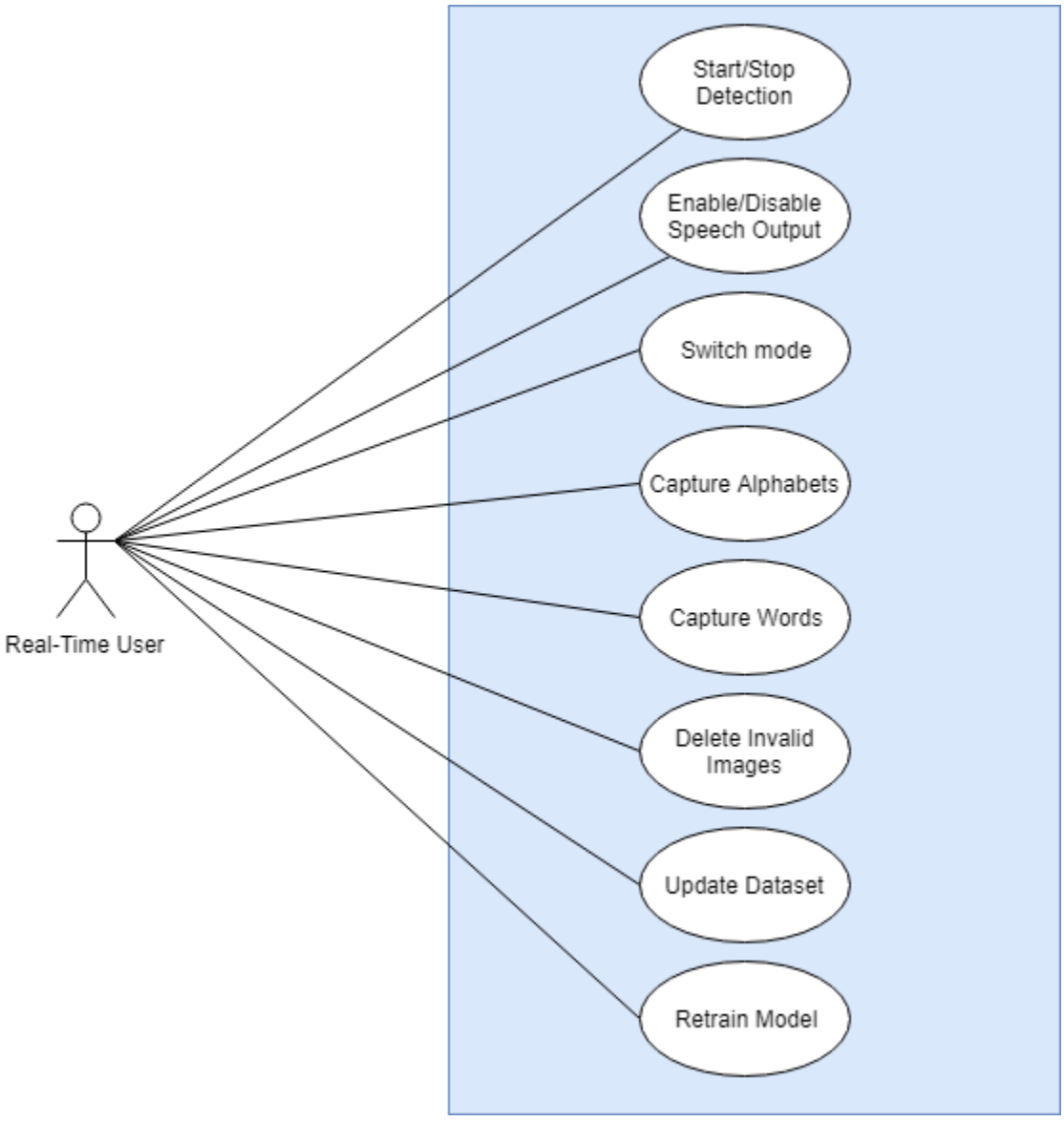


Figure 3.1. (a) Use Case Diagram (Level 1)

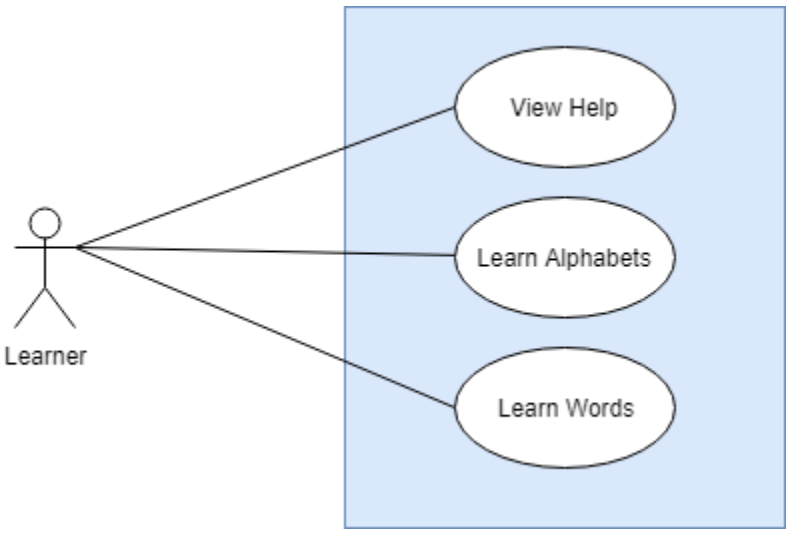


Figure 3.1. (b) Use Case Diagram (Level 1)

3.2. USE CASE DIAGRAMS-(LEVEL-2)

UC_01 (START/STOP DETECTION)

TABLE 3.1. UC_01 (START/STOP DETECTION)

Use Case ID	UC_01	
Use Case Name	Start/Stop Detection	
Description	Click the icon to Start or Stop Detection	
Primary Actor	Real-time User	
Secondary Actor	None	
Pre-Condition	Recognition Stopped / Recognition Started	
Post-Condition	Outputs Word/Alphabet / Outputs nothing	
Basic Flow	Actor Action	System Action
	Click Start/Stop Icon to Start/Stop Detection	System will Start/Stop Detecting gestures
Alternate Flow	If user performs invalid gesture system will not output anything.	

Figure 3.2. UC_01 (Start/ Stop Detection)

UC_02 (LEARN ALPHABETS)

TABLE 3.2. UC_02 (LEARN ALPHABETS)

Use Case ID	UC_02	
Use Case Name	Learn Alphabets	
Description	Click Learn Icon to and then click Alphabets to Start Learning Process	
Primary Actor	Learner User	
Secondary Actor	None	
Pre-Condition	User should be at Main window	
Post-Condition	Click Start Learning then Click alphabets to Start process and skip for the next Alphabet	
Basic Flow	Actor Action	System Action
	Click Learn Icon, Click Alphabet then Perform Gestures, Skip for next Alphabet	Start Alphabet Learning Process, Output Good for matched gesture and Bad for unmatched gesture
Alternate Flow	None	

Figure 3.3. UC_02 (Learn Alphabets)

UC_03 (ENABLE/DISABLE SPEECH OUTPUT)

TABLE 3.3. UC_03 (ENABLE/DISABLE SPEECH OUTPUT)

Use Case ID	UC_03	
Use Case Name	Enable/Disable Speech Output	
Description	Enables and Disables Speech Output	
Primary Actor	Real-time User	
Secondary Actor	None	
Pre-Condition	Speech checkbox should be checked/unchecked	
Post-Condition	Outputs Speech / nothing	
Basic Flow	Actor Action	System Action
	User Checks/Unchecks the box to enable or disable speech.	System outputs Speech if enabled or nothing if disabled.
Alternate Flow	None	

Figure 3.4. UC_03 (ENABLE/DISABLE SPEECH OUTPUT)

UC_04 (SWITCH MODE)

TABLE 3.4. UC_04 (SWITCH MODE)

Use Case ID	UC_04	
Use Case Name	Switch Mode	
Description	Switches between different modes	
Primary Actor	Real-time User	
Secondary Actor	None	
Pre-Condition	Some mode should be selected	
Post-Condition	Mode should start its process	
Basic Flow	Actor Action	System Action
	User clicks on the Mode icon to start that Mode	System starts the process of selected Mode
Alternate Flow	User can go back from selected mode and select some other mode as well.	

Figure 3.5. UC_04 (SWITCH MODE)

UC_05 (VIEW HELP)

TABLE 3.5. UC_05 (VIEW HELP)

Use Case ID	UC_05	
Use Case Name	View Help	
Description	It is user manual which helps the user about the software	
Primary Actor	Real-Time User	
Secondary Actor	None	
Pre-Condition	Should be at the Main window	
Post-Condition	Should open User manual	
Basic Flow	Actor Action	System Action
	Click Help "?" Icon to show user manual window	System should show Help window
Alternate Flow	None	

Figure 3.6. UC_05 (VIEW HELP)

UC_06 (LEARN WORDS)

TABLE 3.6. UC_06 (LEARN WORDS)

Use Case ID	UC_06	
Use Case Name	Learn Words	
Description	Click Learn Icon to and then click Words to Start Learning Process	
Primary Actor	Learner User	
Secondary Actor	None	
Pre-Condition	User should be at Main window	
Post-Condition	Click Start Learning then Click alphabets to Start process and skip for the next Words	
Basic Flow	Actor Action	System Action
	Click Learn Icon, Click Words then Perform Gestures, Skip for next Word	Start Word Learning Process, Output Good for matched gesture and Bad for unmatched gesture
Alternate Flow	None	

Figure 3.7. UC_06 (LEARN WORDS)

UC_07 (CAPTURE ALPHABETS)

TABLE 3.7 UC_07 (CAPTURE ALPHABETS)

Use Case ID	UC_07	
Use Case Name	Capture Alphabets	
Description	This Process will Capture Alphabets	
Primary Actor	Real-time User	
Secondary Actor	None	
Pre-Condition	Should be in Capture Dataset Window	
Post-Condition	It should show the images	
Basic Flow	Actor Action	System Action
	User should perform gestures of alphabets	System should capture the dataset
Alternate Flow	None	

Figure 3.8. UC_07 (Capture Alphabets)

UC_08 (CAPTURE WORDS)

TABLE 3.8. UC_08 (CAPTURE WORDS)

Use Case ID	UC_08	
Use Case Name	Capture Words	
Description	This Process will Capture Words	
Primary Actor	Real-time User	
Secondary Actor	None	
Pre-Condition	Should be in Capture Dataset Window	
Post-Condition	It should show the images	
Basic Flow	Actor Action	System Action
	User should perform gestures of Words	System should capture the dataset
Alternate Flow	None	

Figure 3.9. UC_08 (Capture Words)

UC_09 (DELETE INVALID IMAGES)

TABLE 3.9. UC_09 (DELETE INVALID IMAGES)

Use Case ID	UC_09	
Use Case Name	Delete Invalid Images	
Description	This process will Delete Invalid Images captured by the user	
Primary Actor	Real-time User	
Secondary Actor	None	
Pre-Condition	Should Capture the gestures	
Post-Condition	Images should be deleted from the folder	
Basic Flow	Actor Action	System Action
	Capture the gestures and select the one which are invalid and should be deleted	System should delete the invalid images from the folder
Alternate Flow	None	

Figure 3.10. UC_09 (Delete Invalid Images)

UC_10 (UPDATE DATASET)

TABLE 3.10. UC_10 (UPDATE DATASET)

Use Case ID	UC_10	
Use Case Name	Update Dataset	
Description	This process should Update the Dataset	
Primary Actor	Real-time User	
Secondary Actor	None	
Pre-Condition	User should perform Gesture	
Post-Condition	It should be added in the database for training	
Basic Flow	Actor Action	System Action
	User should perform the gesture. Delete the invalid images and store the rest in database	System should update its dataset with the new images
Alternate Flow	None	

Figure 3.11. UC_10 (Update Dataset)

UC_11 (RETRAIN MODEL)

TABLE 3.11. UC_11 (RETRAIN MODEL)

Use Case ID	UC_11	
Use Case Name	Retrain Model	
Description	This feature Retrains Model after adding images in dataset	
Primary Actor	Real-time User	
Secondary Actor	None	
Pre-Condition	User should capture images	
Post-Condition	System should be able to detect the gesture	
Basic Flow	Actor Action	System Action
	Save the images in the database and then retrain the model.	System should be able to detect the gesture.
Alternate Flow	None	

Figure 3.12. UC_11 (Retrain Model)

Chapter 4: Design

4.1. ARCHITECTURE DIAGRAM

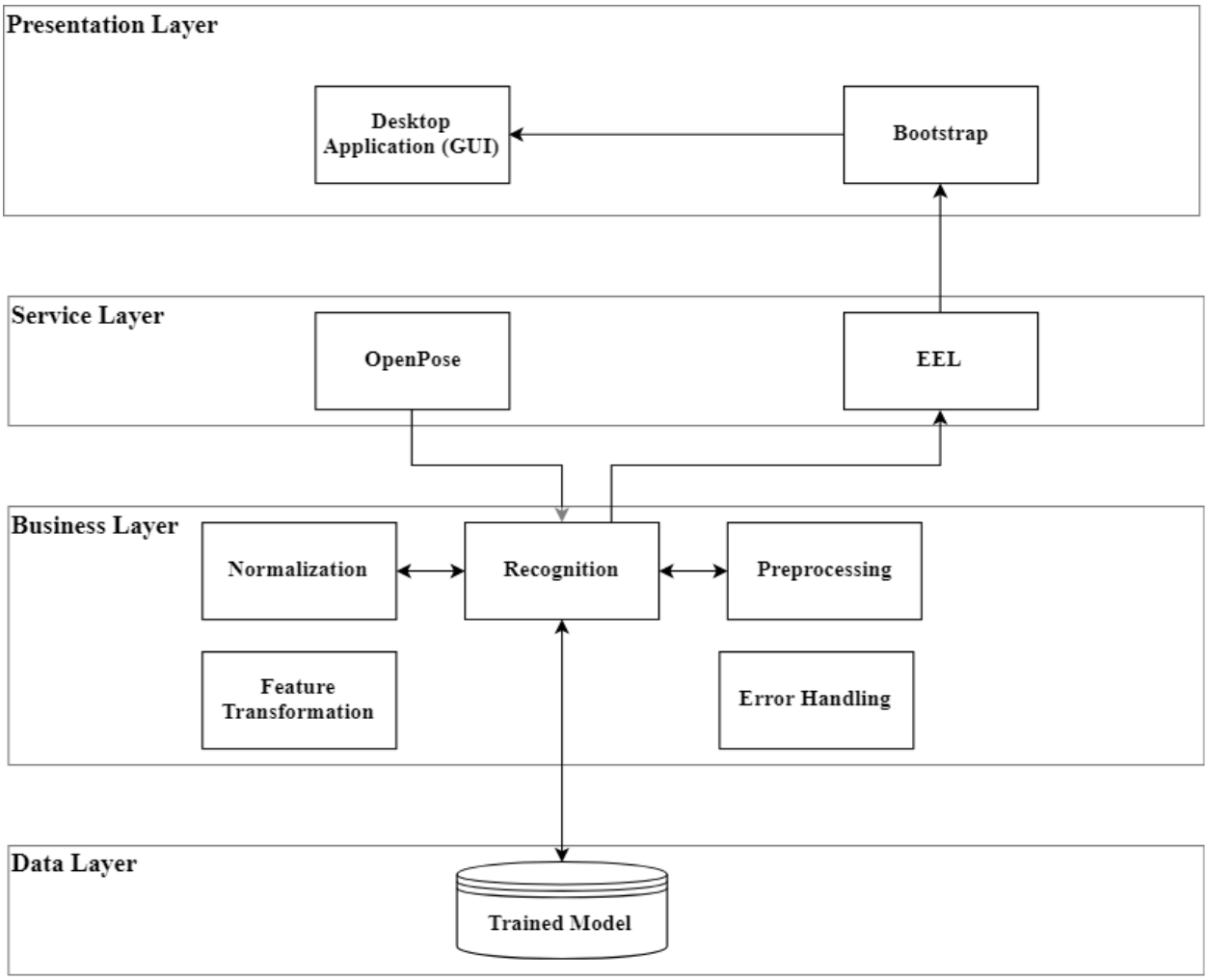


Figure 4.1. ARCHITECTURE DIAGRAM

4.2. DATA FLOW DIAGRAM

LEVEL 0

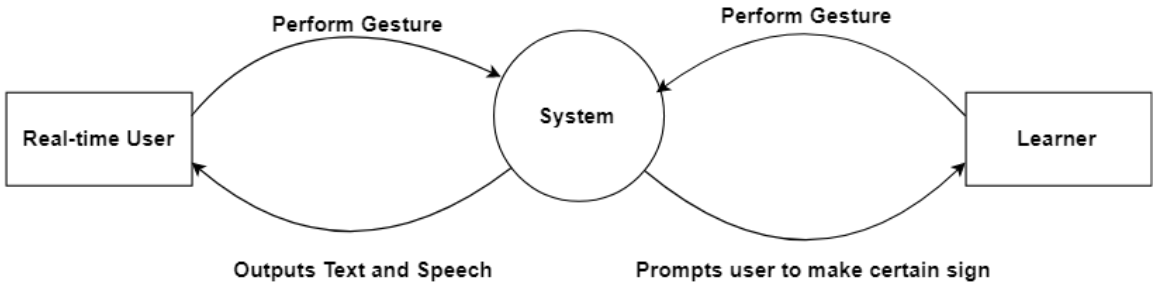


Figure 4.2. DATA FLOW DIAGRAM (LEVEL 0)

LEVEL 1

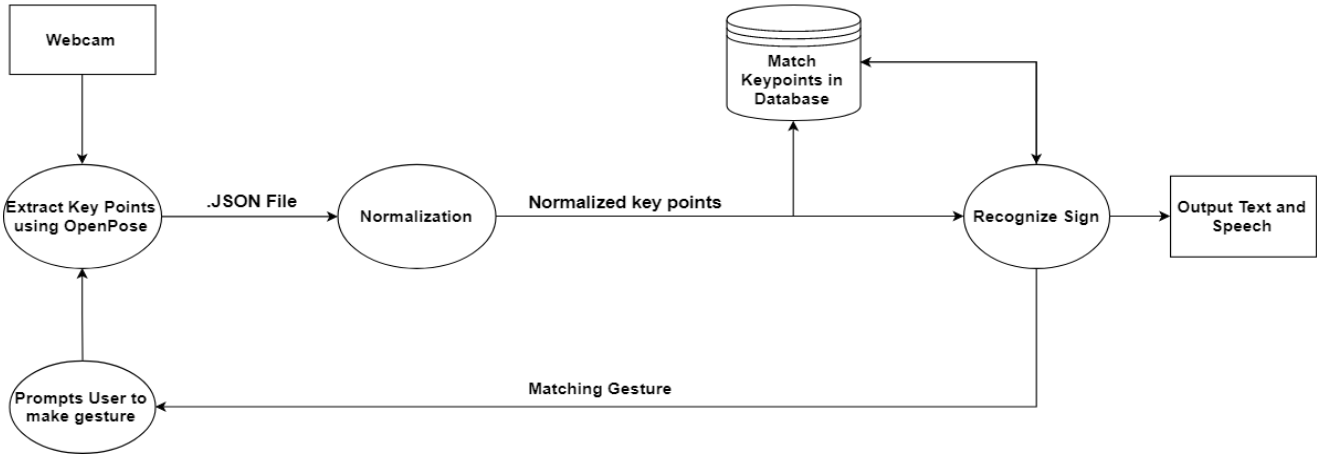


Figure 4.3. DATA FLOW DIAGRAM (LEVEL 1)

4.3. ACTIVITY DIAGRAMS

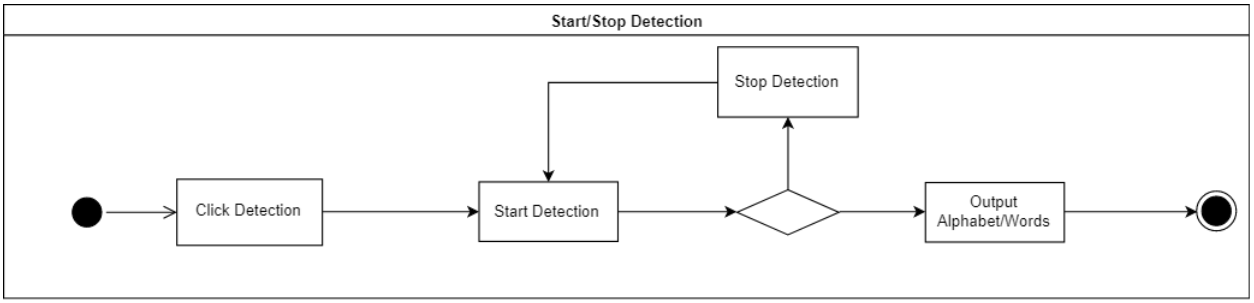


Figure 4.4 Activity Diagram (Star/Stop detection)

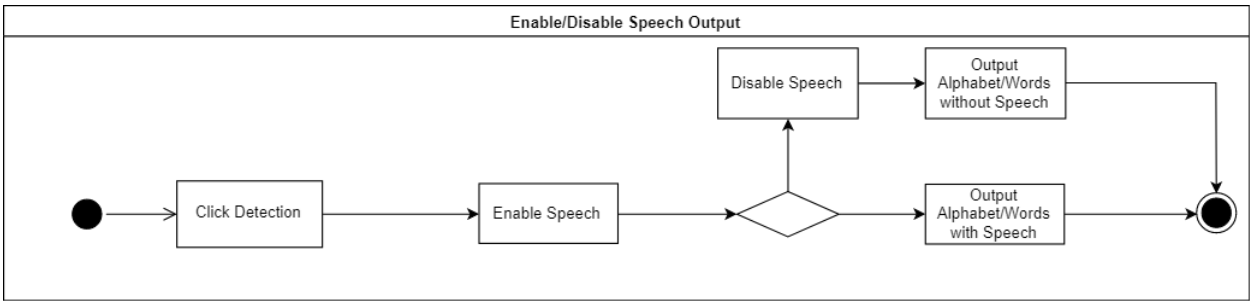


Figure 4.5 Activity Diagram (Enable/Disable Speech)

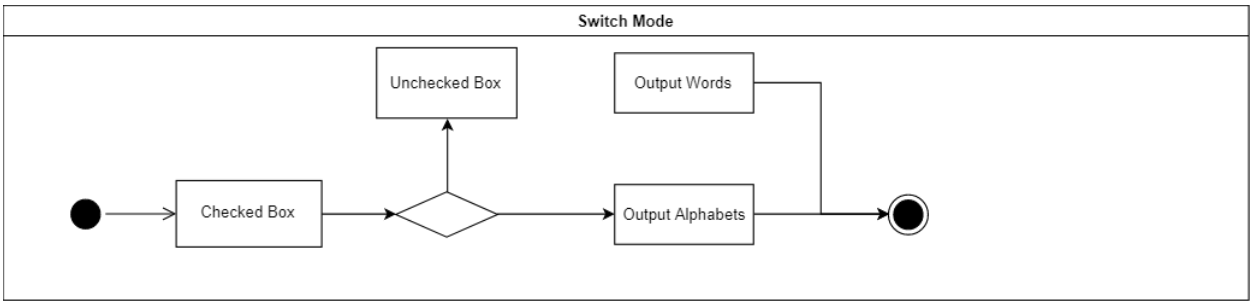


Figure 4.6. Activity Diagram (Switch Mode)

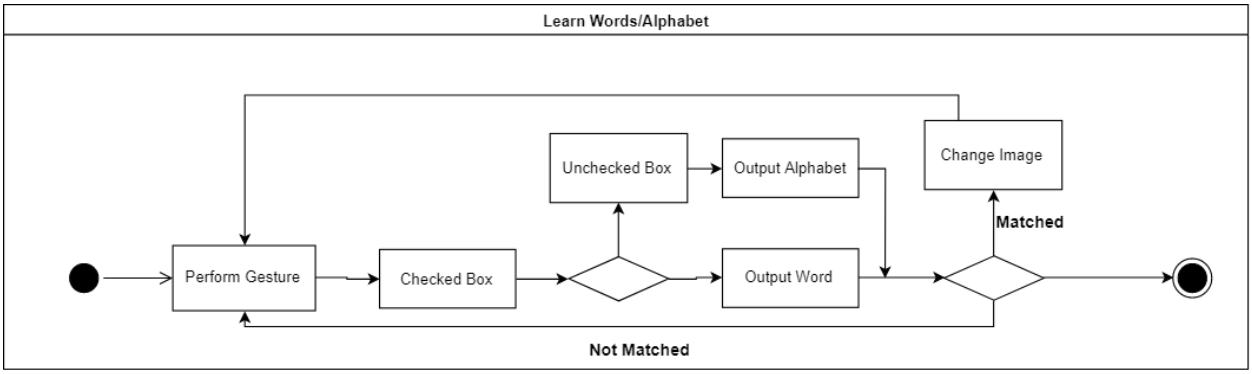


Figure 4.7. Activity Diagram (Learn Word/Alphabet)

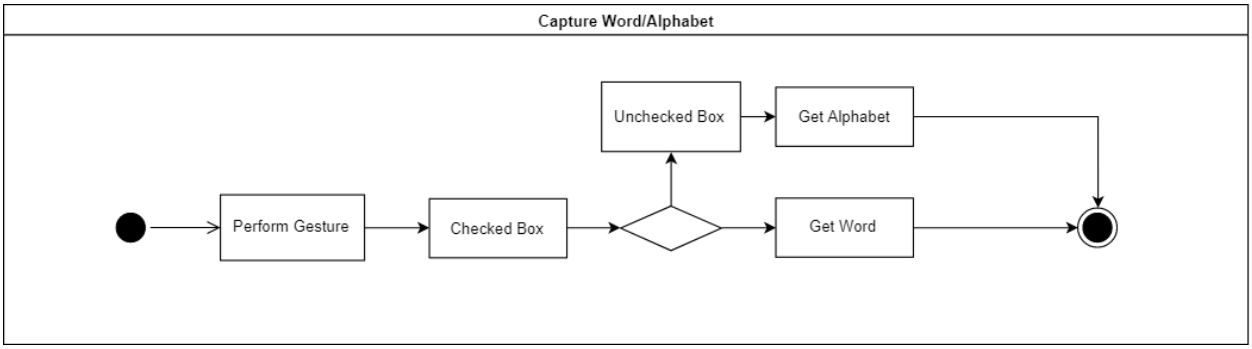


Figure 4.8. Activity Diagram (Capture Word/Alphabet)

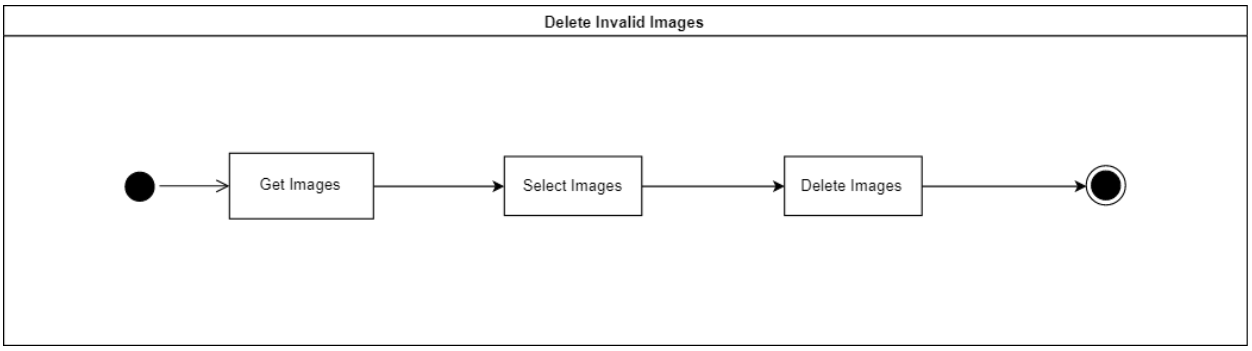


Figure 4.9. Activity Diagram (Delete Invalid Images)

4.4. SEQUENCE DIAGRAMS

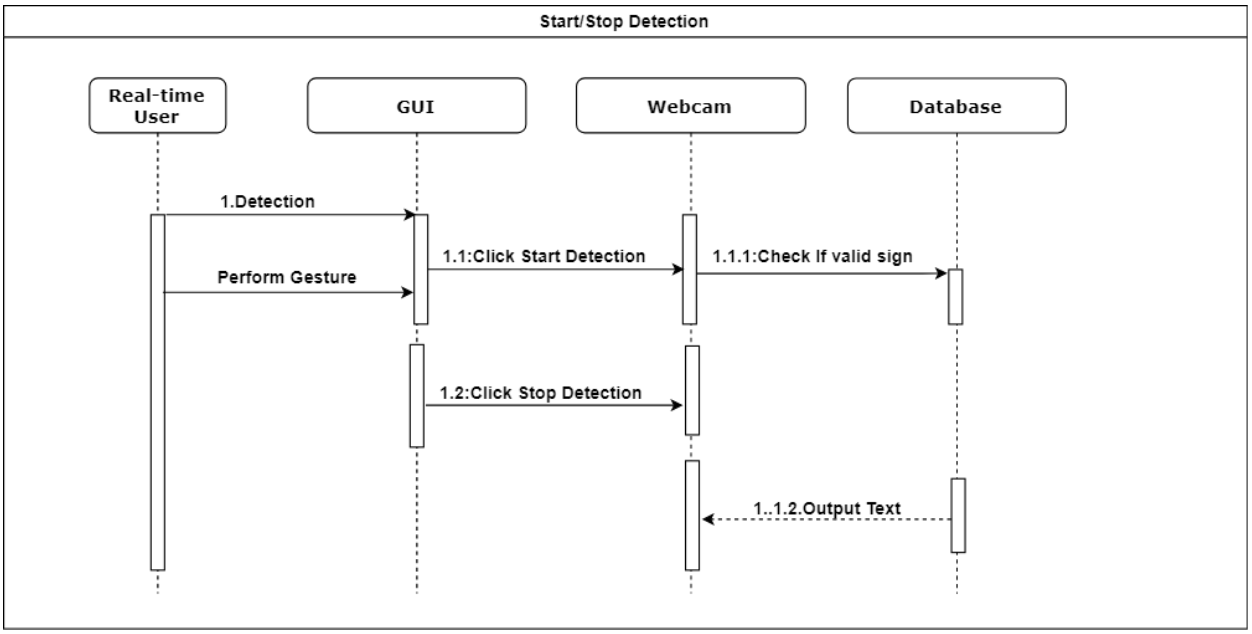


Figure 4.10 Sequence Diagram (Start/Stop Detection)

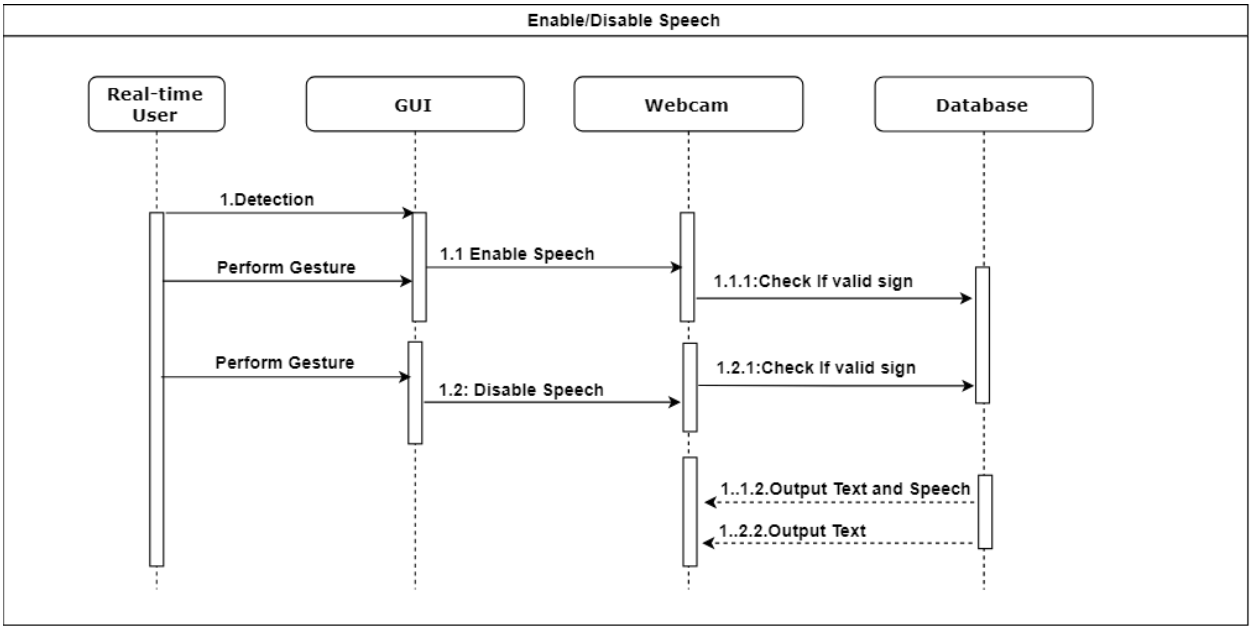


Figure 4.11. Sequence Diagram (Enable/disable Speech)

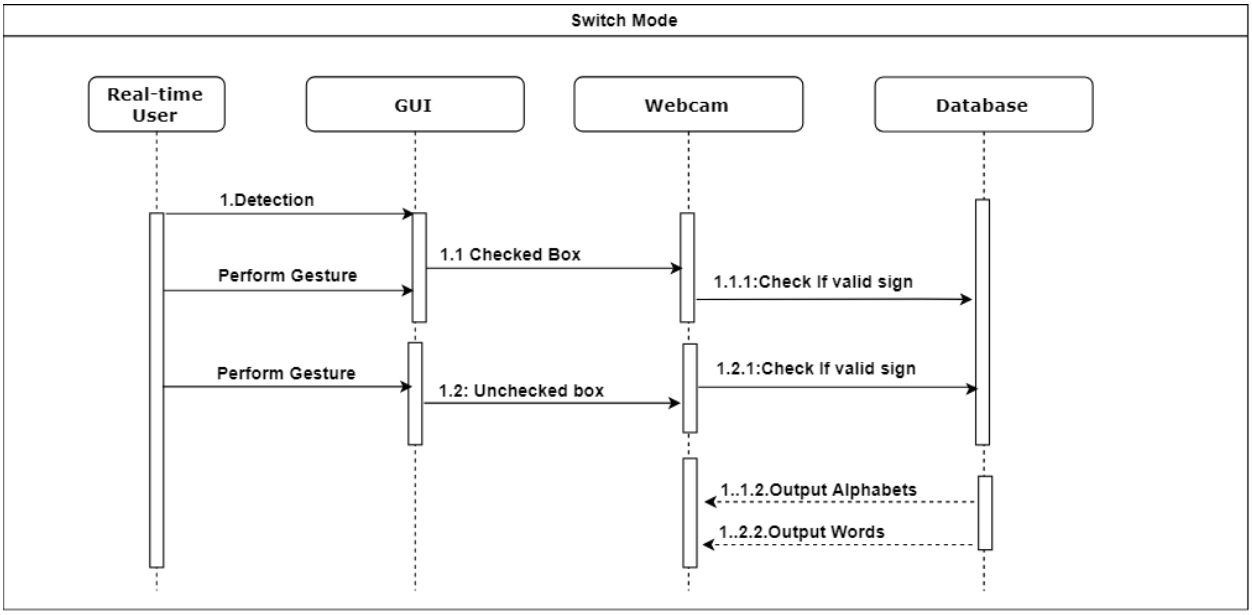


Figure 4.12 Sequence Diagram (Switch Mode)

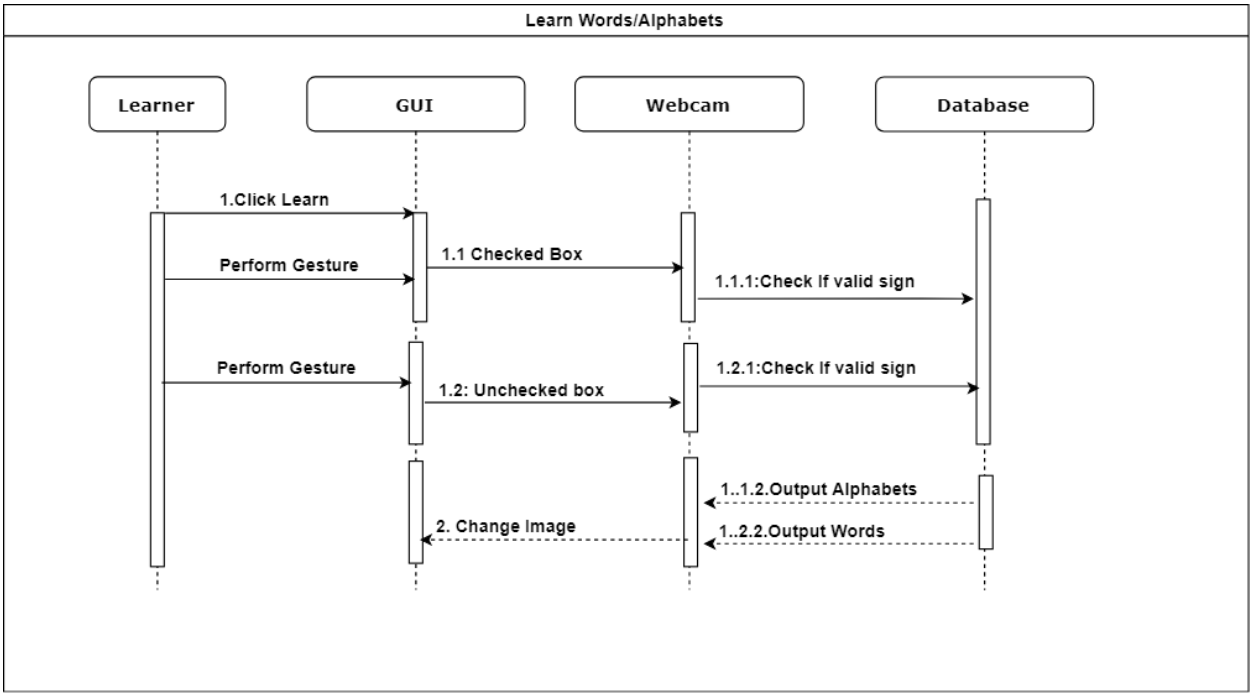


Figure 4.13 Sequence Diagram (Learn Words/Alphabets)

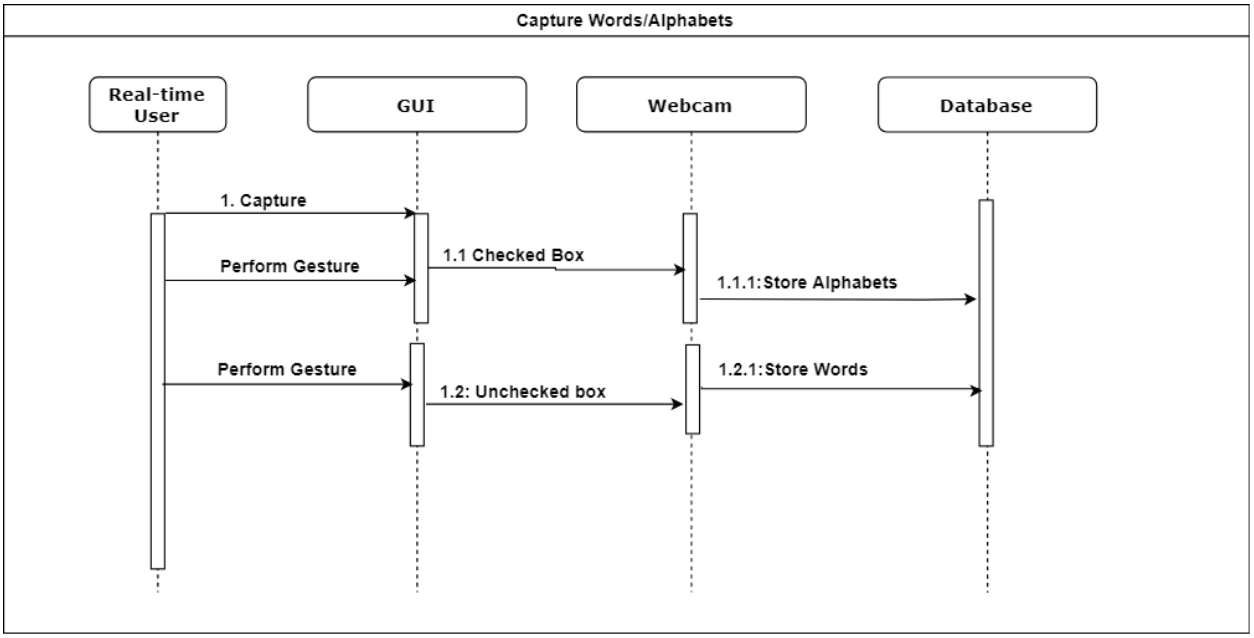


Figure 4.15 Sequence Diagram (Capture Words/Alphabets)

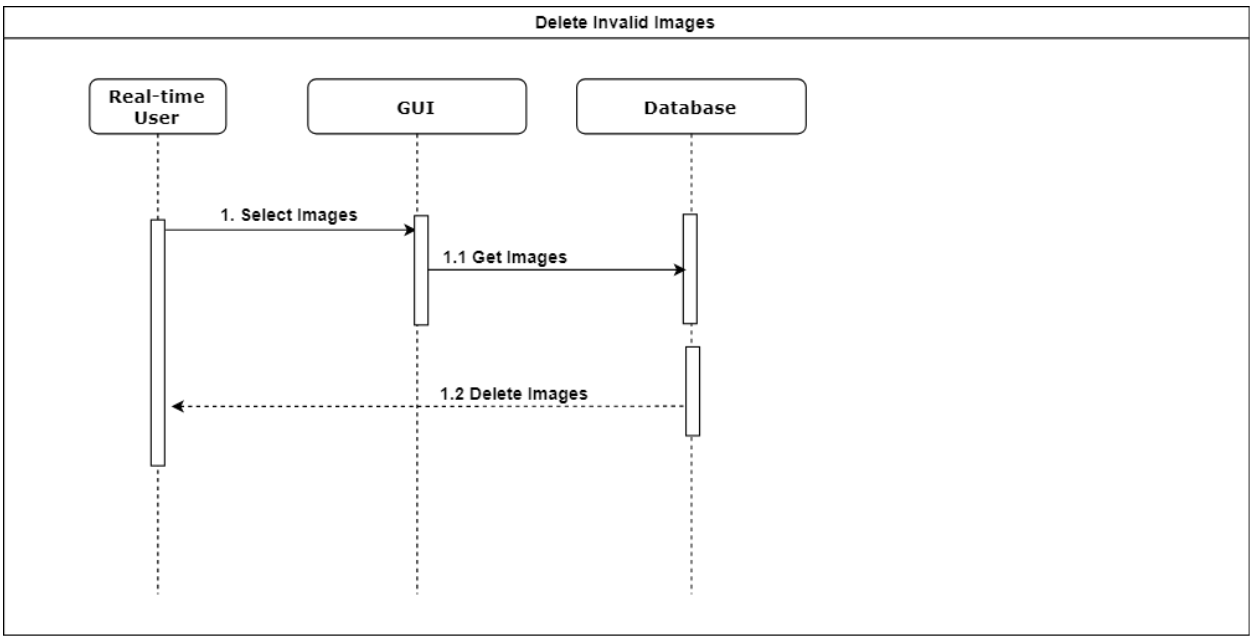


Figure 4.16. Sequence Diagram (Delete Invalid Images)

4.5. COLLABORATION DIAGRAM

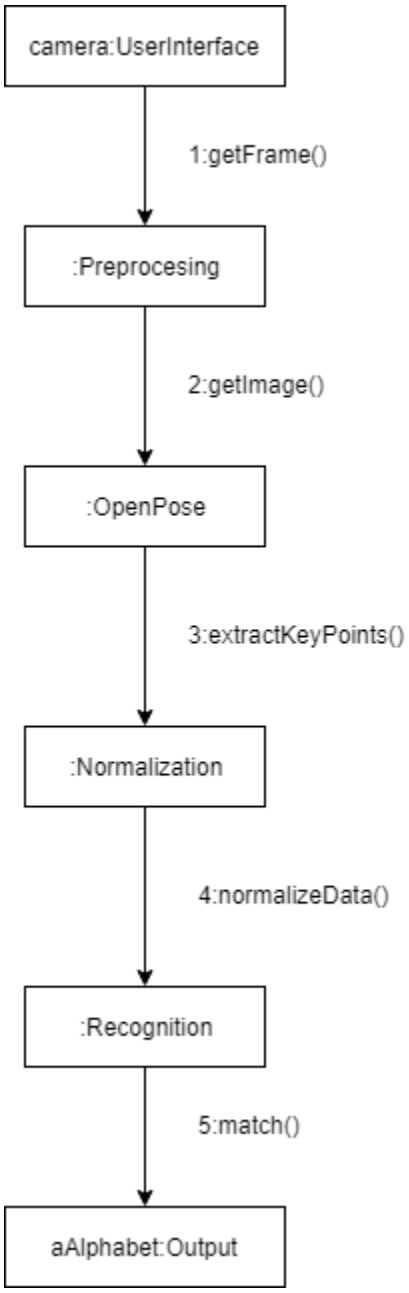


Figure 4.17. Collaboration Diagram

4.6. STATE TRANSITION DIAGRAM

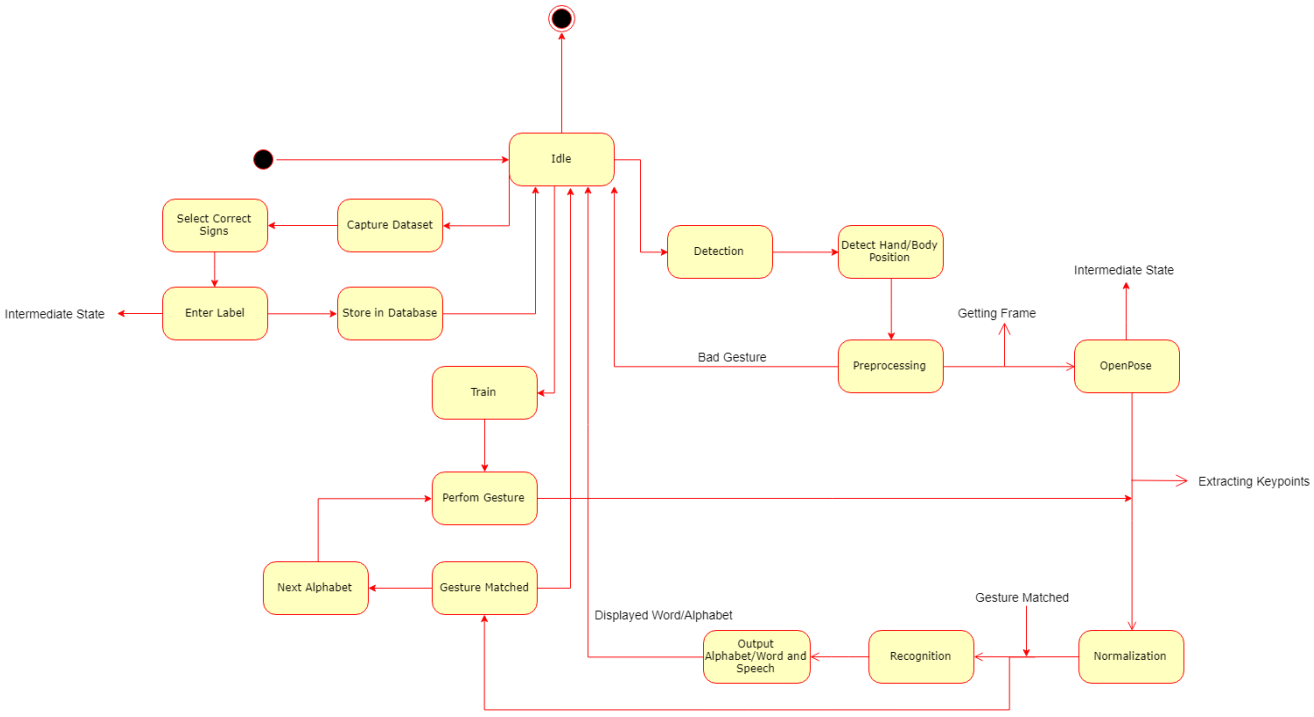


Figure 4.18. State Transition Diagram

4.7. COMPONENT DIAGRAM

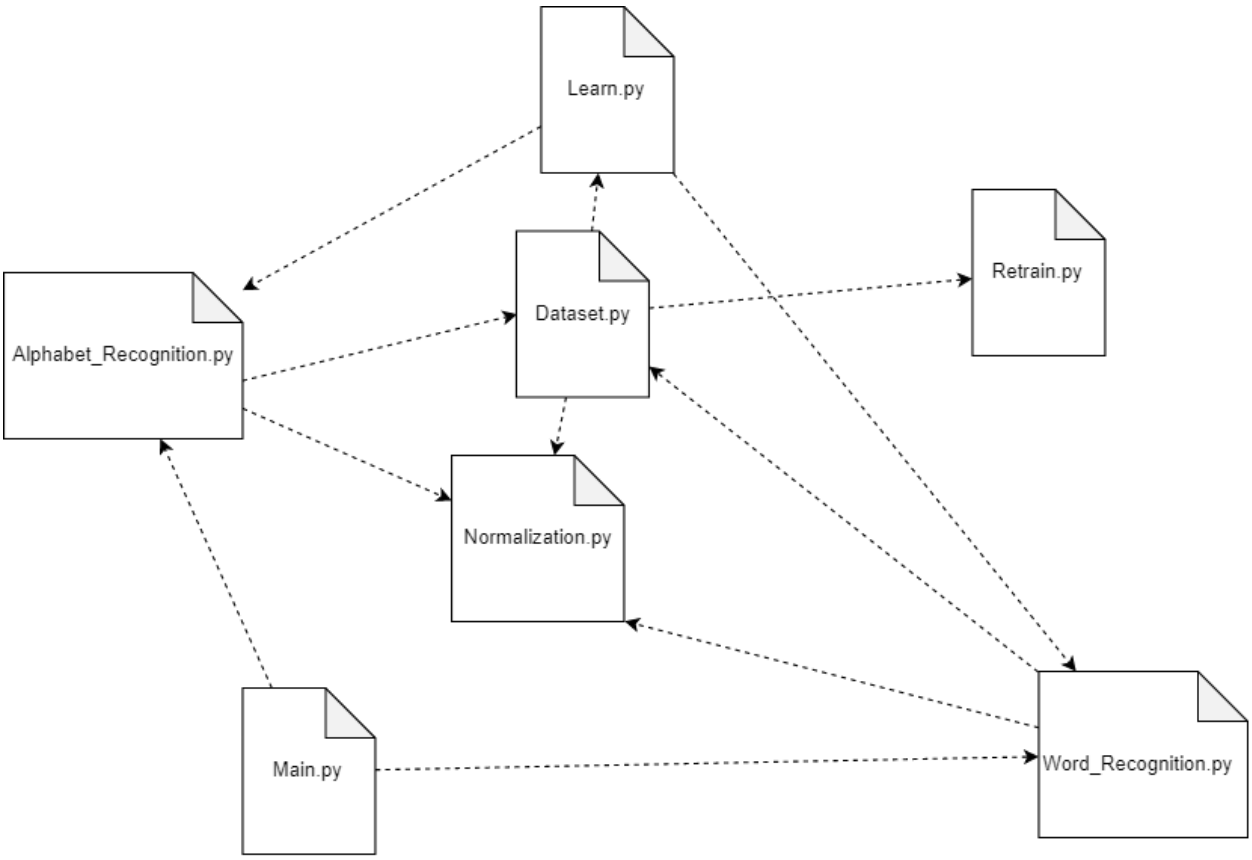


Figure 4.19. Component Diagram

4.8. DEPLOYMENT DIAGRAM

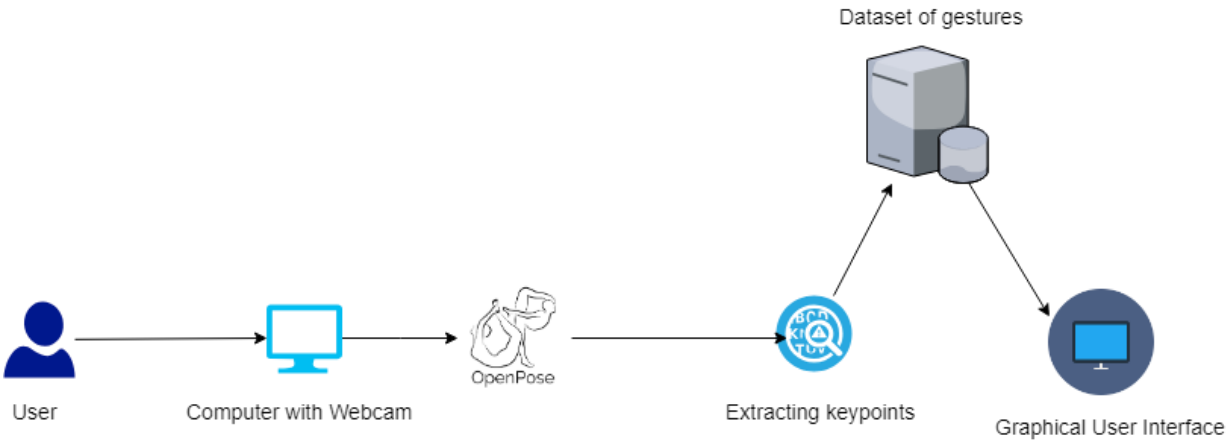


Figure 4.20. Deployment Diagram

CHAPTER 5: TESTING

5.1. TEST CASE SPECIFICATIONS

5.1.1. START DETECTION

Positive Test Case	
ID	TC_START_DETECTION_SUCCESS
Priority	High
Description	To start detecting signs made by user
Reference	FR_01
Users	Real time user
Pre-requisites	• Recognition is not already started
Steps	• Open the application • Go to detection page • Click ‘Start Detection’ button
Input	Button Click
Expected result	recognition started successfully
Status	Tested, passed.

Negative Test Case	
ID	TC_START_DETECTION_FAILURE
Priority	High
Description	To start detecting signs made by user
Reference	FR_01
Users	Real time user
Pre-requisites	• Recognition is not already started
Steps	• Click ‘Start Detection’ button
Input	Button Click
Expected result	does not allow recognition to be restarted
Status	Tested, passed.

5.1.2. RECOGNIZE SIGN

Positive Test Case	
ID	TC_RECOGNIZE _SUCCESS
Priority	High
Description	Recognize the sign made by user and convert it into text and speech
Reference	FR_02
Users	Real time user
Pre-requisites	• Recognition has started
Steps	• Open the application • Go to detection page • Click ‘Start Detection’ button • Make a valid sign in front of the webcam
Input	Valid sign in front of the webcam
Expected result	Correct conversion into text and speech
Status	Tested, passed.

Negative Test Case	
ID	TC_RECOGNIZE _FAILURE
Priority	High
Description	To start detecting signs made by user
Reference	FR_02
Users	Real time user
Pre-requisites	● Recognition has started
Steps	● Open the application ● Go to detection page ● Click ‘Start Detection’ button ● Make an invalid sign in front of the webcam
Input	Invalid sign in front of the webcam
Expected result	Gives no confidence error
Status	Tested, passed.

5.1.3. STOP DETECTION

Positive Test Case	
ID	TC_STOP_DETECTION_SUCCESS
Priority	High
Description	To stopped detection
Reference	FR_03
Users	Real time user
Pre-requisites	• Recognition has been started
Steps	• Open the application • Go to detection page • Click ‘Stop Detection’ button
Input	Button Click
Expected result	recognition stopped successfully
Status	Tested, passed.

Negative Test Case	
ID	TC_STOP_DETECTION_FAILURE
Priority	High
Description	To stop detection
Reference	FR_03
Users	Real time user
Pre-requisites	● Recognition has been started
Steps	● Click ‘Stop Detection’ button
Input	Button Click
Expected result	do nothing
Status	Tested, passed.

5.1.4. ENABLE SPEECH

Positive Test Case	
ID	TC_ENABLE_SPEECH_SUCCESS
Priority	High
Description	Enable speech output
Reference	FR_04
Users	Real time user
Pre-requisites	● ‘enable speech’ checkbox unchecked
Steps	● Open the application ● Go to detection page ● check the ‘enable speech’ checkbox
Input	check the ‘enable speech’ checkbox
Expected result	speech output enabled successfully
Status	Tested, passed.

5.1.5. DISABLE SPEECH

Positive Test Case	
ID	TC_DISABLE_SPEECH_SUCCESS
Priority	High
Description	Disable speech output
Reference	FR_05
Users	Real time user
Pre-requisites	● ‘enable speech’ checkbox checked
Steps	● Open the application ● Go to detection page ● uncheck the ‘enable speech’ checkbox
Input	uncheck the ‘enable speech’ checkbox
Expected result	speech output disabled successfully
Status	Tested, passed.

5.1.6. DISPLAY RECOGNITION OUTPUT

Positive Test Case	
ID	TC_DISPLAY_OUTPUT_SUCCESS
Priority	High
Description	display recognized sign output on screen
Reference	FR_06
Users	Real time user
Pre-requisites	● Recognition has been started ● output received from recognition module
Steps	● Open the application ● Go to detection page ● Click ‘Start Detection’ button ● Make a valid sign in front of the webcam
Input	Output from recognition module
Expected result	Output displayed successfully
Status	Tested, passed.

5.1.7. PLAY SPEECH OUTPUT

Positive Test Case	
ID	TC_PLAY_SPEECH_SUCCESS
Priority	High
Description	play speech of recognized sign
Reference	FR_07
Users	Real time user
Pre-requisites	● Recognition has been started ● output received from recognition module
Steps	● Open the application ● Go to detection page ● Click ‘Start Detection’ button ● Make a valid sign in front of the webcam
Input	Output from recognition module
Expected result	speech played successfully
Status	Tested, passed.

5.1.8. SWITCH TO WORD MODE

Positive Test Case	
ID	TC_SWITCH_WORD_MODE_SUCCESS
Priority	High
Description	Switch to word mode from alphabet mode
Reference	FR_08
Users	Real time user
Pre-requisites	● ‘switch to word mode’ checkbox unchecked
Steps	● Open the application ● Go to detection page ● check the ‘switch to word mode’ checkbox
Input	check the ‘switch to word mode’ checkbox
Expected result	Switched to word mode from alphabet mode successfully
Status	Tested, passed.

5.1.9. SWITCH TO ALPHABET MODE

Positive Test Case	
ID	TC_SWITCH_ALPHABET_MODE_SUCCESS
Priority	High
Description	Switch to alphabet mode from word mode
Reference	FR_09
Users	Real time user
Pre-requisites	• 'switch to word mode' checkbox checked
Steps	• Open the application • Go to detection page • uncheck the 'switch to word mode' checkbox
Input	uncheck the 'switch to word mode' checkbox
Expected result	Switched to alphabet mode from word mode successfully
Status	Tested, passed.

5.1.10. VIEW HELP

Positive Test Case	
ID	TC_VIEW_HELP_SUCCESS
Priority	Medium
Description	View Help Document
Reference	FR_10
Users	Real time user, Learner
Pre-requisites	• nothing
Steps	• Go to main page
Input	Click Help Button
Expected result	Help document shown in new page successfully
Status	Tested, passed.

5.1.11. DISPLAY SIGN IMAGE

Positive Test Case	
ID	TC_DISPLAY_IMAGE_SUCCESS
Priority	HIGH
Description	display image of specific sign on screen and prompt user to try making that sign
Reference	FR_11
Users	Learner
Pre-requisites	• nothing
Steps	• Open the application • Go to learning page
Input	Learning Module
Expected result	Image displayed successfully
Status	Tested, passed.

5.1.12. VALIDATE SIGN

Positive Test Case	
ID	TC_VALIDATE_SIGN_SUCCESS
Priority	HIGH
Description	Validate sign made by user against an image shown to him
Reference	FR_12
Users	Learner
Pre-requisites	• Webcam working.
Steps	• Open the application • Go to learning page • Make sign shown on screen in front of webcam.
Input	Valid sign made by user in front of webcam.
Expected result	Success message displayed
Status	Tested, passed.

Negative Test Case	
ID	TC_VALIDATE_SIGN_FALIURE
Priority	HIGH
Description	Validate sign made by user against an image shown to him
Reference	FR_12
Users	Learner
Pre-requisites	• Webcam working.
Steps	• Open the application • Go to learning page • Make sign shown on screen in front of webcam.
Input	Invalid sign made by user in front of webcam.
Expected result	Try again message displayed
Status	Tested, passed.

5.1.13. CHANGE SIGN IMAGE

Positive Test Case	
ID	TC_CHANGE_IMAGE_SUCCESS
Priority	HIGH
Description	Show next sign image to user if previous sign was validated
Reference	FR_12
Users	Learner
Pre-requisites	• previous sign was validated
Steps	• Open the application • Go to learning page • Make a valid sign
Input	Validation function
Expected result	Image Changed successfully
Status	Tested, passed.

Negative Test Case	
ID	TC_CHANGE_IMAGE_FALIURE
Priority	HIGH
Description	Show next sign image to user if previous sign was validated
Reference	FR_13
Users	Learner
Pre-requisites	• previous sign was validated
Steps	• Open the application • Go to learning page • Make a valid sign
Input	Validation function
Expected result	Image Changed successfully
Status	Tested, passed.

5.1.14. SKIP SIGN

Positive Test Case	
ID	TC_SKIP_SIGN_SUCCESS
Priority	HIGH
Description	Skip to next sign if user clicks skip button
Reference	FR_14
Users	Learner
Pre-requisites	• sign has not been validated
Steps	• Open the application • Go to learning page • Click ‘Skip’ Button
Input	Click ‘Skip’ Button
Expected result	• Image Changed successfully • moved to next sign
Status	Tested, passed.

5.1.15. SELECT CAPTURE MODE

Positive Test Case	
ID	TC_SWITCH_CAPTURE_MODE_SUCCESS
Priority	High
Description	Select either word mode or alphabet mode from ‘select mode’ dropdown
Reference	FR_15
Users	Real time user
Pre-requisites	● Capture has not been started
Steps	● Open the application ● Go to Capture page ● ‘select mode’ dropdown
Input	‘select mode’ dropdown
Expected result	mode changed to selected dropdown value
Status	Tested, passed.

5.1.16. Select Capture Time

Positive Test Case	
ID	TC_SET_TIME_SUCCESS
Priority	High
Description	Enter capture time in seconds, in ‘enter time’ text field to set time for capture process
Reference	FR_16
Users	user
Pre-requisites	• Capture has not been started
Steps	• Open the application • Go to Capture page • ‘enter time’ text field
Input	‘enter time’ text field
Expected result	time set successfully
Status	Tested, passed.

5.1.17. Display Captured Images

Positive Test Case	
ID	TC_DISPLAY_IMAGES_SUCCESS
Priority	High
Description	Display images captured in capture process
Reference	FR_17
Users	Real time user
Pre-requisites	• some images received from capture process
Steps	• Go to Capture page • Strat Capture
Input	images captured in capture process
Expected result	images displayed on screen
Status	Tested, passed.

5.1.18. DELETE INVALID IMAGES

Positive Test Case	
ID	TC_DELETE_IMAGES_SUCCESS
Priority	High
Description	Select images of invalid signs and delete them
Reference	FR_18
Users	Real time user
Pre-requisites	• some images selected to delete
Steps	• Go to Capture page • Strat Capture • select images
Input	select images
Expected result	images deleted successfully
Status	Tested, passed.

5.1.19. POPULATE DATASET

Positive Test Case	
ID	TC_POPULATE_DATASET_SUCCESS
Priority	High
Description	Click ‘populate dataset’ button to update the dataset.
Reference	FR_20
Users	Real time user
Pre-requisites	• some images captured in capture process
Steps	• Go to Capture page • Strat Capture • Click ‘populate dataset’ button
Input	Click ‘populate dataset’ button
Expected result	dataset updated with new captured files
Status	Tested, passed.

5.1.20. RETRAIN MODEL

Positive Test Case	
ID	TC_TRAIN_MODEL_SUCCESS
Priority	High
Description	Click ‘Train Model’ button to train ANN model with updated dataset
Reference	FR_21
Users	Real time user
Pre-requisites	• no requirements
Steps	• Go to Capture page • Strat Capture • Click ‘Train Model’ button
Input	Click ‘Train Model’ button
Expected result	ANN model trained with updated dataset
Status	Tested, passed.

5.2. Black Box Test Cases

5.2.1. Equivalence Partitions (EP)

Start detection

Description	Using this test case, the sign detection process can be checked		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will start detecting signs and displaying alphabets.	1. Start Openpose. 2. Start Detection.	1. Start Openpose.	It will not detect signs and display alphabets.

Speech checking

Description	Using this test case, the speech of sign can be checked		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will speak the alphabet which you made a sign of.	Make a sign of any alphabet if it exists in the database.	Make any sign randomly.	It will speak of an alphabet which is close to that sign.

STOP DETECTION

Description	Using this test case, the sign detection process can be checked		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will stop detecting signs and displaying alphabets.	1. Start Openpose. 2. Start Detection. 3. Press “Esc” to exit from Openpose.	1. Start Openpose. 2. Start Detection.	It will not stop detecting signs and displaying alphabets.

DETECTION

Description	Using this test case, the detection process can be checked		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will perform Detection function.	1. Select Detection Tab.	1. Select Capture Tab. 2. Select Learn Tab.	It will perform the Capture/Training function.

LEARNING

Description	Using this test case, the Learning process can be checked		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will perform the Leaning function.	1. Select Learning Tab.	1. Select Capture Tab. 2. Select Detection Tab.	It will perform the Capture/Detection function.

CAPTURE

Description	Using this test case, the Capture process can be checked		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will perform the Capture function.	1. Select Capture Tab.	1. Select Detection Tab. 2. Select Learn Tab.	It will perform Detection/Learning function.

HELP

Description	Using this test case, the Help process can be checked		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will guide you about the functions.	1. Select "?" Tab For help.	1. Select Detection Tab. 2. Select Learn Tab. 3. Select Capture Tab.	It will not guide you about the functions instead it will do the function.

CAPTURE DATASET

Description	Using this test case, the Capturing dataset process can be checked.		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will start Capturing Dataset.	1. Insert the seconds you want to run the process.	1. Start without inserting seconds. 2. Insert the alphabet instead of numbers.	It will not start the Capturing Dataset process.

Label Assigning

Description	Using this test case, the Label Assignment process in Capturing Dataset can be checked.		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will store/delete the captured dataset with the label you assigned.	1. Starts Capturing process. 2. Press 'Y' for storing captured dataset 'N' for deleting captured dataset.	1. Starts Capturing process. 2. Press something other than 'Y' or 'N'	It will not store/delete the captured dataset with the label you assigned.

Learning

Description	Using this test case, the Learning process can be checked.		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will start Showing you sign images for you to learn to make them correctly.	1. Select Learn Tab. 2. Click 'Start Learning'	1. Select Learn Tab.	It will not start the process of learning.

Skipping Sign

Description	Using this test case, the Skipping Sign in Learning Process can be checked.		
Valid Output	**Valid Input**	**Invalid Input**	**Invalid Output**
It will Skip the current sign and will show you the next sign.	1. Select Learn Tab. 2. Click 'Start Learning' 3. Skip Sign.	1. Select Learn Tab. 2. Click 'Start Learning'	It will not Skip any sign until you make it correct.

5.2.2. USE CASE TESTING

We have tested the system functionalities against each use case.

5.3. WHITE BOX TEST CASES

We have tested internal code of every component and it was checked that the code written is efficient in utilizing various resources of the system like memory or the utilizing of input and output. The tests were automated using pyTest.

5.3.1. CYCLOMATIC COMPLEXITY

For testing cyclomatic complexity of each function we have used Radon 2.4.0 and flake8. The functions with cyclomatic complexity greater than 5 have been simplified.

5.4. PERFORMANCE TESTING

For testing performance, we have used cProfile a python profiler that provides a set of statistics that describes how often and for how long various parts of the program executed.

5.5. SYSTEM TESTING

After testing each unit we have checked the integrated components as a whole and looked for possible problems, which could have arisen after the integration

5.6. REGRESSION TESTING

In Regression testing, the software was tested against the boundary conditions. Various input fields were tested against abnormal values and it was tested that the software does not behave abnormally at any time.

CHAPTER 6: TOOLS AND TECHNIQUES

6.1. LANGUAGES

1. For Back-End, we have used Python
2. For Front-End we have used:
    a. HTML
    b. CSS
    c. JavaScript

6.2. APPLICATIONS AND TOOLS

- We have used Spyder IDE for backend development due to its highly interactive environment.
- For Front-End we have used Dreamweaver

6.3. LIBRARIES AND EXTENSIONS

- OpenPose Library for extracting skeletal key points
- Eel
- Bootstrap
- jQuery

CHAPTER 7: SUMMARY AND CONCLUSION

7.1. SUMMARY

A noticeable amount of deaf community exists in Pakistan. This system is developed to aid these people by converting Pakistan Sign Language (PSL) into Urdu text and speech. The main objective is to facilitate a large population of hearing-impaired persons and making them an integral part of society. Worldwide efforts have been made to aid the deaf community in communicating with non-signers but most of the existing systems either use specialized sensors or has low performance. The proposed system works by taking images of a user making a particular sign through webcam and uses skeletal tracking combined with machine learning to detect what sign is made by the person.

Currently, there is no publicly available dataset for Pakistan Sign Language. So we decided to make our own dataset for PSL; for that purpose we made an automated dataset capturing system that captures images of user for specified time, extract the key points from images using OpenPose, deletes the images in which OpenPose correctly detects less than 10 key points from hand, plot the skeleton of remaining images and show them to the user, the user then selects the plotted images that does not match the sign he was making and deletes them. The user is then asked to enter the label for remaining images then these images are stored by the system in a folder named as the value of label entered by user.

We have collected the data of 37 Urdu alphabets from 9 different people and 12 Urdu words from 4 different people. We have extracted skeletal key points of each data point in the form of JSON files by using OpenPose. In the case of alphabets, only the 21 key points of the right hand were selected and normalized by scaling and moving them to the center while keeping the original shape of the hand. For Urdu language words, 13 key points from upper body were selected along with 42 key points of right and left hand, theses key points were normalized by scaling the body and each hand, and by moving the body to center, left hand to left wrist point and right hand to right wrist point, while keeping the original shape of the body and the hands. In sign language, the same sign can differ minutely from one instance to another even when it is the same person making the sign. For minimize rotation and positional variance of signs, we decided to synthesize new data from the original data. So, in case of alphabets, we rotated the right hand to +20 and -20 degrees and store them as 2 different positions of same sign hence minimizing rotation variance that existed in the alphabet signs. In the case of words the hands were moved 80 pixels up, down, left and right while keeping the natural posture of arms using Inverse Kinematics(IK) and rotating the hands to +20 and -20 degrees and storing them as 12 different positions of the same signs hence minimizing rotation and positional variance.

We trained two Artificial Neural Network (ANN) models one for alphabet recognition and one for word recognition. To avoid overfitting because of the limited data we considered a simpler ANN model, that performed surprisingly well for both words and alphabets. In Urdu alphabet

recognition model we fed 42 (21x and 21y) input array into the first layer of ANN, following this we feed into hidden layer with 64 units, finally we send everything through the output layer with sigmoid activation, resulting in a 36-element vector. For Urdu word recognition, the same model was used but the input layer was fed with 110 (42+42+26) input array. The alphabet recognition model achieved 99.4% accuracy on test data and ~93% accuracy on real-time inputs through webcam, whereas the word recognition model has achieved 99.2% accuracy on test data and ~88% accuracy on real-time inputs through webcam. Due to the positional variance of signs made by different users the accuracy of real-time input is less than the test data.

We have also introduced an interactive PSL learning system. This system works by showing the user different images of PSL signs and prompting him/her to make that sign, the system then checks whether the user have made the correct sign or not and displays a message accordingly.

7.2. CONCLUSION

The overall PSL translating system is split into 3 main modules, detection module, learning module and capture module. The detection module has two modes one for alphabet detection and other for word detection. The learning module is used for interactive PSL learning and the capture module is used for adding new data to dataset.

Due to lack of large pre-existing PSL dataset we have introduced a new PSL dataset that consists of human body key points of 9 different people making signs of 37 Urdu alphabets and 4 different people making signs of 12 Urdu words. As a result, the dataset contains 4500 Urdu alphabet samples and 2000 Urdu word samples.

Different machine learning models were considered but we found ANN to be most suitable for this task. We were surprised to find that the optimal ANN design came from a simple architecture rather than a complicated one. The final alphabet model design consists of an input layer with 42 units, a hidden layer with 64 units and an output layer with sigmoid activation resulting in 36-element vector. The word recognition model contains an input layer with 110 units a hidden layer with 64 units and an output layer resulting in 12-element vector. The alphabet recognition model achieved 99.4% accuracy on test data and ~93% accuracy on real-time inputs through webcam, whereas the word recognition model has achieved 99.2% accuracy on test data and ~88% accuracy on real-time inputs.

Chapter 8: User Manual

8.1. Launching the Application

To launch the application double, click signDetect.exe file.

8.2. Using the Main Menu

Choose one of the following options from the main menu:

8.2.1. Detection

Go to Detection Page to start Pakistan Sign Language conversion to text and speech.

8.2.2. Learn

Start learning Pakistan Sign Language interactively.

8.2.3. Capture

Go to capture page to capture new data for enhancing the conversion process

8.2.4. Help

Get help from the help document

8.3. Sign Language Detection

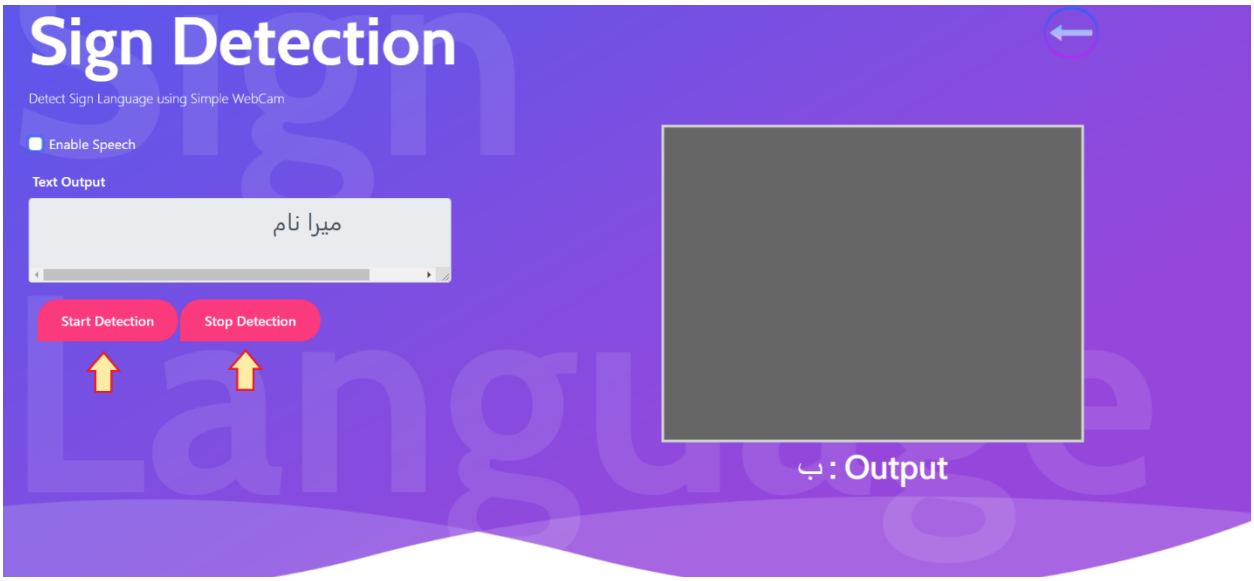
To get to Sign Language Detection, Select Detection from the main menu.

8.3.1. Enabling Speech Output

To Enable speech output check ‘Enable Speech’ checkbox.

8.3.2. START AND STOP DETECTION

1. To Start Detection, click ‘Start Detection Button’
2. To Stop Detection, click ‘Stop Detection Button’



8.3.3. GO BACK

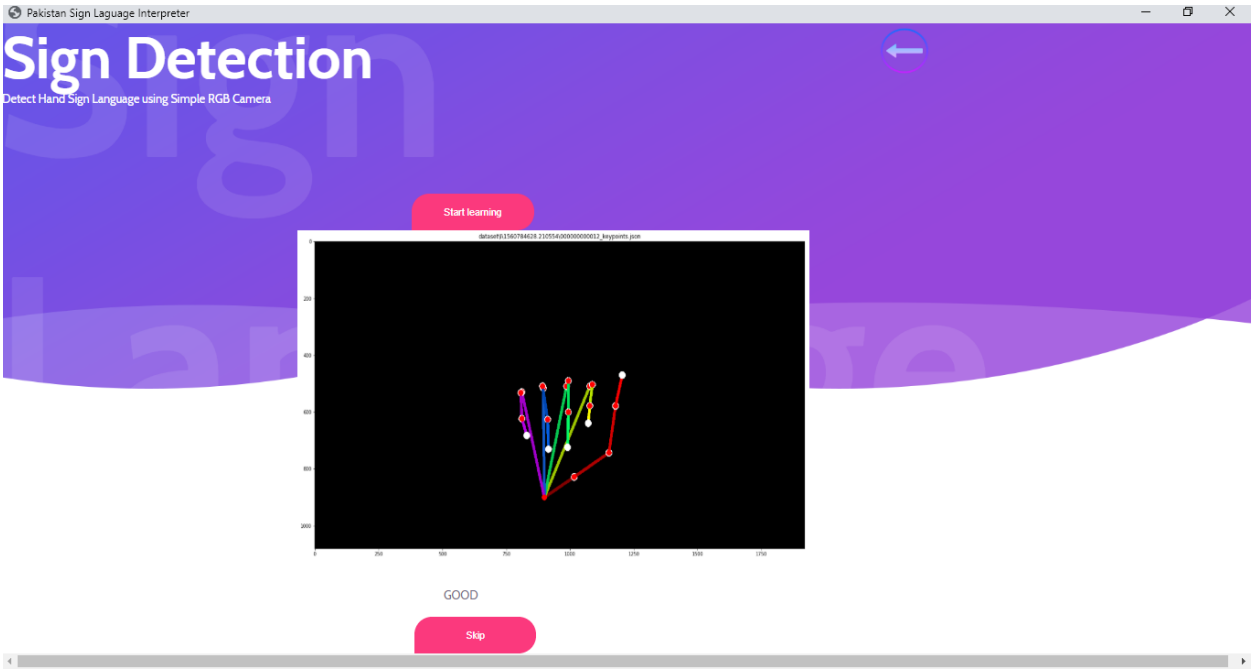
To go back to the main menu, click the back button.

8.4. Learning Sign Language

To get to the learning page, select Learn from the main menu.

8.4.1. Start Learning

To start learning click ‘Start Learning’ button



8.4.2. Skip Sign

To skip the current displayed sign, click ‘Skip’ button

8.5. Capture New Data

To get to capture page, select Capture from the main menu.

8.5.1. Start Capturing

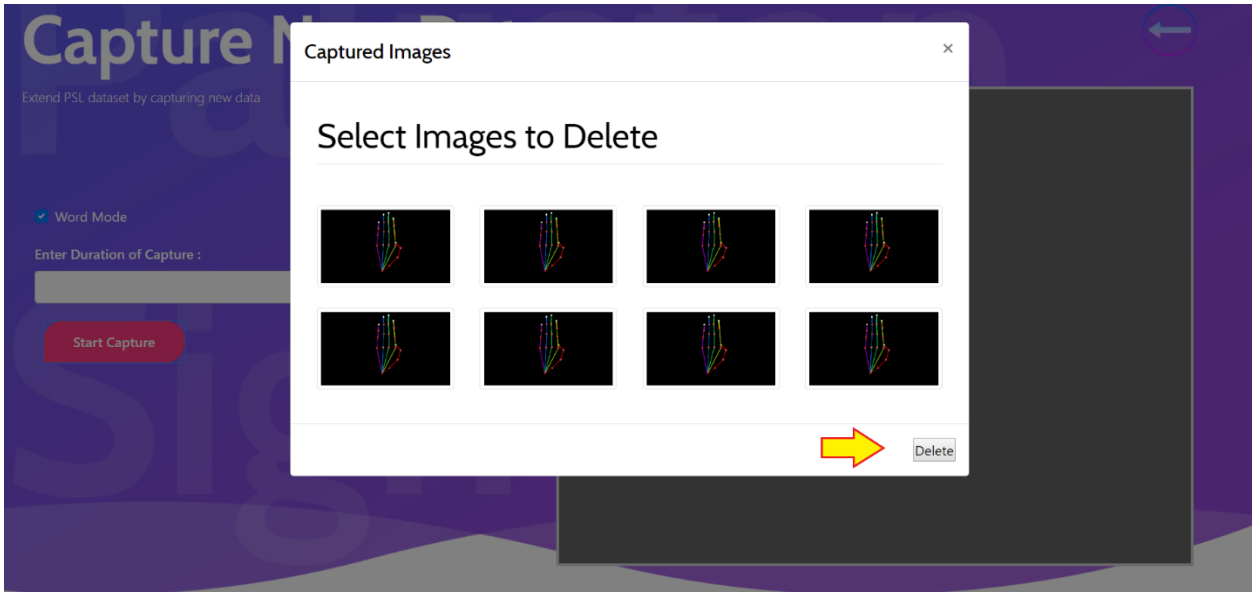
To Start Capturing:

1. Select either word mode or alphabet mode from dropdown.
2. Enter time capture process should run for in seconds.
3. click ‘Start’ button.
4. perform a specific PSL sign.

8.5.2. DELETE INVALID IMAGES AND SET THE LABEL

To delete invalid images:

1. Select images you want to delete.
2. enter a label in the text field.
3. click ‘Go’ button.



8.5.3. Populate And Retrain

To update the dataset and retrain ANN model:

1. check if images left after deletion are all valid.
2. click ‘populate’ button to update the dataset.
3. click ‘retrain’ button to train the ANN model.
4. if images are not valid click ‘discard’ button

CHAPTER 9: LESSONS LEARNT AND FUTURE ENHANCEMENTS

As Pakistan Sign Language contains thousands of words it is not possible to make a comprehensive system that covers all these signs at this level. We have provided a system that can convert all Urdu alphabets and some Urdu words into text and speech and have also provided a framework for adding more words to the system. The proposed system has given very high accuracy rate for PSL recognition. But there is still room for improvement. Following guidelines are suggested for the future enhancements:

- The system can be further extended to allow recognition of dynamic PSL signs.
- The dataset for PSL can be extended further for more vocabulary.
- Natural Language Processing (NLP) techniques can be used to formulate meaningful sentences from PSL words.
- The text and speech conversion to PSL animation can be added to allow two way communication between signer and non-signer.

REFERENCES

1. Statistics, D. (2015). Deaf Statistics. Pakistan Association of Deaf. Retrieved 24 October 2018, from http://padeaf.org/deaf-statistic/

2. Waqar, K. (2014). Disability: Situation in Pakistan (1st ed.). Karachi: Right to Education Pakistan. Retrieved from http://www.itacec.org/document/gaw/gaw2014/2.%20Disability%20Pages%202.pdf

3. Rini Akmeliawatil, Melanie PO-Leen Ooi, et al," RealTime Malaysian Sign Language Translation using Color Segmentation and Neural Network. Instrumentation and Measurement Technology Conference Warsaw, Poland.IEEE.1-6, 2007.

4. Nicholas Born, "Senior Project Sign Language Glove", ELECTRICAL ENGINEERING DEPARTMENT. California Polytechnic State University 1-49, 2010.

5. M. Wasim, Abdulbasit Shaikh, A.A Siddiqui. Communicator for Hearing-Impaired Persons using Pakistan Sign Language (PSL). The International Journal of Advanced Computer Science and Applications, Vol. 9, No. 5, 2018.

6. Kausar, S., Javed, M. Y., & Sohail, S. Recognition of gestures in Pakistani sign language using fuzzy classifier. In Proceedings of the 8th conference on Signal processing, computational geometry and artificial vision (pp. 101-105). World Scientific and Engineering Academy and Society (WSEAS), August, 2008.

7. Ali, S. A., "Detection of Urdu Sign Language using Harr Algorithms", International Journal of Inventive Engineering and Sciences (IJIES) ISSN: 2319–9598, Volume-1, Issue-6, May 2013.

8. Zhe Cao, Yaser Sheikh, et al, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", arXiv:1611.08050v2 [cs.CV], Apr 2017.

9. Avinash Navlani," Support Vector Machines with Scikit-learn", available at https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python, 12 July 2018.

10. Avinash Navlani," KNN Classification using Scikit-learn", available at https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn, 2 Aug 2018.

11. Yun-lei Cai et al "A KNN Research Paper Classification Method Based on Shared Nearest Neighbor", Proceedings of NTCIR-8 Workshop Meeting, Tokyo, Japan, June 2010.

12. Ms. Sonali. B. Maind et al," Research Paper on Basic of Artificial Neural Network", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 2 Issue: 1, 2014.

13. Mart´ın Abadi, Ashish Agarwal, et al," TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", Preliminary White Paper, November 9, 2015.