MIAN AFZAAL ZAHOOR
01-134182-094
SULEMAN
01-134182-059

Urdu Sign Language Recognition

Bachelor of Science in Computer Science

Supervisor: Ms. Maria Mahmood

Department of Computer Science
Bahria University, Islamabad

June 2022

Certificate

We accept the work contained in the report titled "URDU SIGN LANGUAGE RECOGNITION", written by MIAN AFZAAL ZAHOOR AND SULEMAN as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Ms. Maria Mahmood (Lecturer)

_____

Internal Examiner: Name of the Internal Examiner (Title)

_____

External Examiner: Name of the External Examiner (Title)

_____

Project Coordinator: Dr. Moazam Ali (Assistant Professor)

_____

Head of the Department: Dr. Arif ur Rahman (Sr. Associate Professor)

_____

June 22^{nd}, 2022

Abstract

USLR stands for Urdu Sign Language Recognition. Communication is a primary need of humans, and to communicate, the medium of language is used. Most people in the world can listen and speak while using different languages, while some people who are listening or speaking have to use sign language to convey their message to another person. In Pakistan, where deaf people prefer Urdu sign language as a way to communicate, computer-based USLR interpreters are required.

The purpose of this project is to close the communication gap between the non-signers and signers. The system works by taking the input images of the signer through the camera and displaying the alphabet they made using that hand sign. Our data set contains 37 Urdu language alphabets.

Acknowledgments

In the name of the Almighty ALLAH, the Most Gracious and the Most Merciful. All praises to ALLAH for the qualities and His approval in completing this undertaking. We would also like to express our gratitude and appreciation to our respected supervisor, Ms. MARIA MAHMOOD, who provided her guidance and support at different and complex stages of the project and supervised us in an appropriate and well-defined manner. Moreover, We would also like to acknowledge the support of our parents throughout the course of this project.

MIAN AFZAAL ZAHOOR, SULEMAN
Islamabad, Pakistan

June 2022

"I've learned that people will forget what you said, people will forget what you did, but people will never forget how you made them feel."

Maya Angelou

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
IDE	Integrated development environment
ML	Machine Learning
PSL	Pakistan Sign Language
RGB	Red Green Blue (Colour Image)
ROI	Region of Interest
UI	User Interface
UML	Unified Modeling Language
USL	Urdu Sign Language
USLR	Urdu Sign Language Recognition

Chapter 1

Introduction

Communication is essential to all human beings, mainly as it allows us to express ourselves, and our emotions and helps in communicating our feelings. We communicate through gestures, speech, body language, writing, reading, and with the help of what we see (visual aids). Among all of these, speech is used as one of the most commonly used mediums for communication. However, for the speaking-impaired minority, there will always be a communication gap because of their disability. For such people, an interpreter or a visual aid is used to deliver their message and communicate.

Figure 1.1 shows the hand signs that are used in the sign language of disabled people. The signs represented are according to IEEE standards.

Figure 1.1: Urdu Sign Language

Sign language mainly uses hand-operated communication to convey the message. Sign

language involves combining hand shapes, orientations, and actions of the hands, arms, or body to show the speaker's thoughts. However, these methods are still a little pricey and time-consuming because they can't be used everywhere and when you need help.

1.1 Objectives

Developed a sensor-less and cheaper Urdu sign language recognition system for dumb people that converts the sign language into the written alphabet for better communication. To achieve the above-mentioned objectives, we have focused on the following provisions:

- Developed an easy-to-use mobile application that anyone with little or no technological knowledge can use.
- A system of translation to assist deaf (mute) people in communicating their message.
- A sign language learning application for those who do not understand sign language.

1.2 Scope

Pakistan has a sizable deaf community. According to a detailed study based on statistical data regarding deaf people in Pakistan, approximately 31 million Pakistanis are suffering from various types of disability, of which 10 million are hearing impaired, accounting for approximately 33 percent of all disables. A critical point is that 55 percent of all disabled people are between the ages of 3 and 18. In addition to CDC statistics, data show that 1 to 3 children out of every 1,000 have hearing loss. Other research has found rates ranging from 2 to 5 per 1,000 children.

So the main motivation for this project is basically to provide a means of communication to those people/children's who cannot speak, a better chance to live in society, and to share their thoughts more easily. We are also making sure to make this project in such a way that it is easier to use so that people with almost zero knowledge of technology can use it. Some methods that were implemented previously couldn't carry out the solutions properly, which left room for improvement in the next projects. Our model will be used to extract the sign, either with noise or without noise. With proper implementation methods, our project will carry out the extraction of signs into written characters. The proposed application will be used to teach toddlers with speaking disability, Urdu haroofs in sign language. Though it may ve extended to recognise words and sentences and act as a complete platform for deaf and dumb people to convey their message.

1.3 Problem Description

Dumb (mute) people use hand signs to communicate, while in society, the need to learn sign language among those who can speak is not very great. So, the mute person faces many problems while communicating as the person in front of them does not understand them. The standard interpreter method is very expensive, and not everyone can afford it. So, to help them overcome this communication gap, there is a need for improvement. As a result, there is a need for a system that easily recognises the various hand signs of sign language and converts them into text in order to convey the message of disabled people to non-disabled people.

Figure 1.2 shows two people talking using the hand signs. The signs being used can changed according to the regions.

Figure 1.2: Communication using Sign Language

1.4 Problem Solution

We have proposed an Urdu Sign Language recognition system based on the human hand gestures that are processed by the library OpenCV. Because it only detects bodies, our recognition system is effective in a variety of settings. The system based on human hand detection works well regardless of the mute person, since then the variance of the signs extracted is negligible. The benefit of this system is that the dumb people will feel it is easy to convey their message, even if the person in front does not understand sign language.

1.5 Methodology

- We have used Python modules to extract and use the image for processing and recognizing hand signs.

- Furthermore, preprocessing steps are being implemented on the desired image, like segmentation, filtering, etc.
- Then we have to done feature extraction for sign detection to detect the sign of the hand using a HSV separation.
- Later, this process is followed by the ML Algorithm. We have compared the image and then find the best-matched result for that sign made from the given data set. Once the sign is matched, the desired alphabet is displayed on the screen.

Figure 1.3 shows the methodology of our project. The main stages followed by the system, while processing the input image.

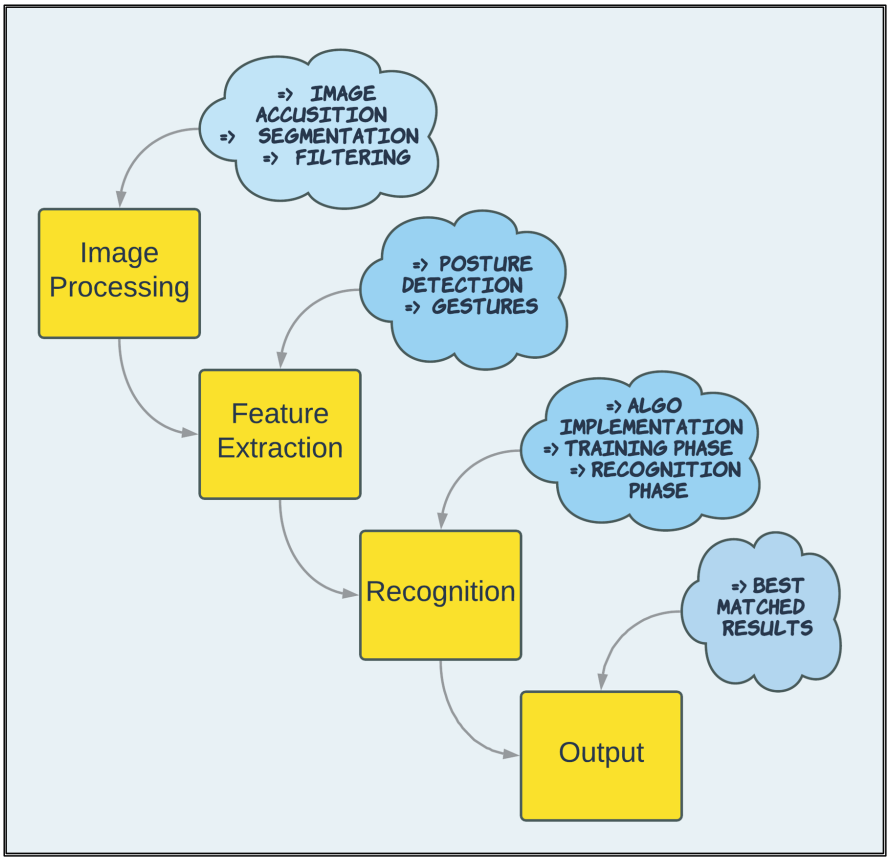

Figure 1.3: Methodology of System

1.6 Tools and Technology

Below is a list of the tools and technologies that are being used in the making of this project. Spyder/Juypter are being used for the development of the back-end of the application, and Android Studio is used for the development of the front-end.

- Jupyter/Spyder
- Python
- Android Studio
- Java

Chapter 2

Literature Review

The literature study is done to get comfortable and distinguish the new strategies utilized in sign language recognition methods, how they are used, and at what stages. This literature study depends on feature extraction, classification of signs, and the techniques used on extracted images. The literature review contains the work of some scientists like Daniel Stein, Dr. Srinath S., and Adnan Abid, who have worked on the relevant idea for English as well as for the Urdu language. The literature review contains the project's information regarding

- Pre-processing techniques used
- The feature upon which the work is done.
- Algorithms used and the accuracy observed.
- Limitations

The challenges and limitations gathered from the literature review provided us with the motivation to work on this idea. This motivation significantly increased the reasons why new and improved systems should be built. Urdu language hasn't been considered of much importance compared to the advancements that have been made in the English language, we chose to work on this language/project.

2.1 Existing Solutions

The implementation of a sign language system in any kind of communication, either verbal or written, has two unique methods: one using data gloves and the other one using image processing. Worldwide, steps have been taken to aid the deaf community in conveying

their message to non-signers. We have taken the points from each of the research papers that are mentioned in the reference column of this literature review.

The collection of data is a critical component of research in all fields of study. While data gathering methods differ according to discipline, the emphasis on ensuring precise and authentic collection leftovers is consistent. Regardless of the discipline of the study, the systematic collection of data is a critical element in ensuring the integrity of the research. Data collection is a critical ceremonial step since it verifies that the gathered data is both precise and accurate.

2.2 Methods for Data Input

Sign language has always been an essential factor in deaf and mute people's communities. People have put a lot of effort into different regions' sign languages in order to make it easier for them to communicate with the rest of the community. As the region changes, so does the speaking language, and with it, the sign language as well.

In the research papers mentioned below, people have used videos and static images to take the input image and further translate it into text and speech. Different approaches have been used to take an input and put it into a system that includes

- Input as a video from which extracted frames are used but with a (2.3 percent) tracking error rate.
- Static images as input through the camera/gallery.

The work has been done in different languages like ASL, Irish Sign Language, and more. However, the work done on Urdu Sign Language is not enough to make it publicly available. That also became the motivation to start working on this project. Some of the data sets that were found in the mentioned research papers are as follows:

- RWTH-Boston-104.
- ASL alphabets gestures.
- Pakistan sign Language alphabets.

2.3 Process Implemented

To translate a hand sign into any oral or written text, the interpretation of that sign must go through some processes to determine what the mute person has said and what to convey to the referring person. Different methods are done to make the correct assumption and the area in which the system works. So basically, to work on the sign, the ROI

(Region of Interest) is extracted from the image, and different preprocessing techniques are implemented on that extracted image, and at the end, it is further utilised to match the data set. Different people have used different methods, as mentioned below.

- To convert the sign made into oral language, the baseline system is extended by the features (including the hand position and hand movement) taken into consideration. Then the alignment algorithm is used to make local decisions (velocity, trajectory, acceleration, etc.) with success. Moreover, the extracted result or matched alphabet is conveyed using the Standardized Statistical Machine Language (SMT). This model uses videos as data input and then the preprocessing techniques are implemented on the extracted frames from the video while using the hand and body position as features for classification.
- In another model, the RGB image is converted to grey scale and later to binary to start processing. After the conversion, the preprocessing (erosion, dilation, hiding irrelevant details, etc.) techniques are carried out. Hand pose is used as the feature that is extracted and processed. Later, that image is used to match the character and display the results. This model uses non-occluded gestures with no noise present in the pictures. This method was also used to identify the numbers along with the alphabet.
- The model used to identify the Pakistani Sign Language uses the primary skin classification filter to extract the hand from the image upon color-based segmentation. This model uses clear background images. Preprocessing techniques convert the RGB image to grey using the Grey World algorithm, and then the image is resized to a standard size, and DWT is applied for feature extraction.

2.4 Learning Algorithms Used

An algorithm is an integral part of the model because it determines the accuracy that the model is getting from the implementation. The accuracy of a model is also determined by if the implemented algorithm is providing the correct results.

The accuracy is determined after the feature extraction and preprocessing techniques are applied to the extracted image. All the algorithms and how well they worked on the model are shown in the table below.

Table 2.1: Comparison Table for different models

Model	**Algorithms**	**Accuracy**
Automatic Sign Language Language to English Translation	RWTH-BOSTON-104	82.4 Percent
American Sign Language Recognition System	Edge Detection	87.5 Percent
A Vision-Based Approach for Pakistan Sign Language Alphabet Recognition	ANN	84.6 Percent

Table 2.1 shows comparison of the algorithms that were used in the previous models along with their final accuracy. Each model have worked on different algorithm and each model shows the acceptable percentage of accuracy.

Each model has its own implementation and reasons for using the preferred algorithm, as the model works in a different way to determine the wanted result.

2.5 Proposed Model

The implemented model for this project have two phases: front-end (mobile application) and back-end (model implementation). The data set that is being used for this model has static binary images. From the input image, the preprocessing techniques are applied to that image, and from the processed image, the features is to be extracted. If matched using the CNN algorithm, the extracted image is used to classify the character. And the written character is displayed on the screen. The front end is has simple view and it only takes static images as input and display the character as output.

Chapter 3

Requirement Specifications

A software requirements specification (SRS) is a document that outlines the functions and performance standards for the software. It also outlines the functionality the product needs to have in order to meet the needs of both business and user stakeholders.

3.1 Functional Requirements

A functional requirement outlines the service that the software must provide. It gives details on a piece of software or a software system. The inputs, behaviours, and outputs of a software system are all referred to as a function.

3.1.1 Hand Detection

This is one of the main functionalities of the system that it should detect the hand from the whole image that is given as input to the system and use it for further processing.

Table 3.1: Functional Requirement for Hand Detection

Identifier	01
Title	Hand Detection
Requirement	The mobile application must recognize the signs made by the user
Source	Supervisor elucidate it
Rationale	Main functionality of our application
Restrictions and Risks	Accuracy
Dependencies	No dependencies
Priority	High

Table 3.1 shows the functional requirement of the application. The above table shows the Hand Detection functionality divided into separate components for better understanding.

3.1.2 Hand Segmentation

This requirement is related to the hand segmentation, that the hand sign should be segmented from the whole image. In this functionality the system is required to successfully segment out the hand from the whole picture.

Table 3.2: Functional Requirement for Hand Segmentation

Identifier	02
Title	Hand Segmentation
Requirement	The system must successfully segment objects from the image
Source	Brainstorming
Rationale	Segmentation is necessary for predicting output
Restrictions and Risks	Noise in image may cause difficulty
Dependencies	No dependencies
Priority	High

Table 3.2 shows the functional requirement of the application. The above table shows the Hand Segmentation functionality divided into separate components for better understanding.

3.1.3 Mode

This requirement is related to the mode, that the user should be provided with the choice to either detect hand using the real image taken at the particular moment using camera or using the image from gallery.

Table 3.3: Functional Requirement for Mode

Identifier	03
Title	Mode
Requirement	The system must take image from camera as well as import from gallery
Source	Brainstorming
Rationale	To give more options to user
Restrictions and Risks	None
Dependencies	No dependencies
Priority	Medium

Table 3.3 shows the functional requirement of the application. The above table shows the Mode functionality divided into separate components for better understanding.

3.1.4 Training and Testing

This requirement is related to the Training and Testing of the data-set, that the images should be compared with the trained data-set to get the better accuracy.

Table 3.4: Functional Requirement for Training and Testing

Identifier	04
Title	Training and Testing
Requirement	Training and Testing data is required to test the system
Source	Brainstorming
Rationale	Training and Testing is required to see if accuracy is achieved
Restrictions and Risks	None
Dependencies	No dependencies
Priority	High

Above 3.4 shows the functional requirement of the application. The above table shows the Training and Testing functionality divided into separate components for better understanding.

3.1.5 Classification

This requirement is related to the classification, that the images should be classified as correct characters. Also that the system should use the same classifier that is being used in the trained model.

Table 3.5: Functional Requirement for Classification

Identifier	05
Title	Classification
Requirement	Classification and Recognition of sign is necessary
Source	Task analysis
Rationale	To correctly identify the sign
Restrictions and Risks	None
Dependencies	No dependencies
Priority	High

Table 3.5 shows the functional requirement of the application. The above table shows the classification functionality divided into separate components for better understanding.

3.1.6 Input format

This requirement is related to the input format, that the input file should be of images. As the system should only deal with images, so the input taken should be in the form of static images.

Table 3.6: Functional Requirement for Input format

Identifier	06
Title	Input format
Requirement	Input could only be an image format
Source	Brainstorming
Rationale	Handling videos as well will complicate things
Restrictions and Risks	System will take longer processing time
Dependencies	No dependencies
Priority	Medium

Table 3.6 shows the functional requirement of the application. The above table shows the Input Format functionality divided into separate components for better understanding.

3.1.7 Output Interface

This requirement is related to the output interface, that the application should have a useful interface that allows the proper implementation of the actions as well as it should also properly display the result on screen that is obtained from the system after processing of the input image.

Table 3.7: Functional Requirement for Output Interface

Identifier	07
Title	Output Interface
Requirement	An interface is required to initiate conversion
Source	Task analysis
Rationale	Output of system is necessary
Restrictions and Risks	None
Dependencies	No dependencies
Priority	High

Table 3.7 shows the functional requirement of the application. The above table shows the Output Interface functionality divided into separate components for better understanding.

3.2 Non-Functional Requirements

System qualities including security, stability, performance, maintainability, scalability, and usability are defined by nonfunctional requirements. They act as limitations or restrictions on how the system is designed for the various backlogs.

3.2.1 Accessibility

This requirement is related to the accessibility of the application, that the application should be downloadable on any android device and that it should also support the functionalities for that current android version.

Table 3.8: Non-Functional Requirement for Accessibility

Identifier	08
Title	Accessibility
Requirement	Any user can download this mobile application
Source	Brainstorming
Rationale	To work when needed
Restrictions and Risks	None
Dependencies	No dependencies
Priority	High

Table 3.8 shows the non-functional requirement of the application. The above table shows the Accessibility functionality divided into separate components for better understanding.

3.2.2 Application Reliability

This requirement is related to the application reliability, that the application should be reliable and should handle all types of images given as input. If any irrelevant image is given as input it should return the proper output rather than crashing the system.

Table 3.9: Non-Functional Requirement for Application Reliability

Identifier	09
Title	Application Reliability
Requirement	The application must work under any system load
Source	Brainstorming
Rationale	To process image of any size
Restrictions and Risks	Heavy size images
Dependencies	No dependencies
Priority	Medium

Table 3.9 shows the nonfunctional requirement of the application. The above table shows the Application Reliability functionality divided into separate components for better understanding.

3.2.3 Response

This requirement is related to the application response, that the application should return the result of the image given as input in the real time. Irrelevant of the size of the image it

should return the response to the user as soon as possible.

Table 3.10: Non-Functional Requirement for Response

Identifier	10
Title	Response
Requirement	Application will display the result in real time
Source	Brainstorming
Rationale	Output processing will be unconventional for user
Restrictions and Risks	Processing time will depend on the size of image
Dependencies	No dependencies
Priority	Medium

Table 3.10 shows the nonfunctional requirement of the application. The above table shows the Response functionality divided into separate components for better understanding.

3.2.4 Learn-ability

This requirement is related to the learn-ability of the hand signs, the application should also provide the easy of learning hand signs as well and also to provide the user with complete manual to operate the application.

Table 3.11: Non-Functional Requirement for Learn-ability

Identifier	11
Title	Learn ability
Requirement	Mobile Application provides simple interfaces for input/output so that all the users either laymen or experts can use it with just basic understanding of English language and computer handling
Source	Brainstorming
Rationale	Make application easy to use for user
Restrictions and Risks	None
Dependencies	No dependencies
Priority	Medium

Table 3.11 shows the nonfunctional requirement of the application. The above table shows the Learn-ability functionality divided into separate components for better understanding.

3.2.5 Maintenance

This requirement is related to the application maintenance, that the application should be easy to maintain and if any bug or error occurs it should be easily removed from the system.

Table 3.12: Non-Functional Requirement for Maintenance

Identifier	12
Title	Maintenance
Requirement	Our system is specified for one task. So, few changes are required with time. Therefore, maintenance will be easier
Source	Task analysis
Rationale	Expansion
Restrictions and Risks	None
Dependencies	No dependencies
Priority	Low

Table 3.12 shows the nonfunctional requirement of the application. The above table shows the Maintenance functionality divided into separate components for better understanding.

3.2.6 Accuracy

This requirement is related to the application accuracy, that the application should accurately detect the hand signs. It also requires that the system should also detect the invalid image and discard these images while providing the message of invalid image to the user.

Table 3.13: Non-Functional Requirement for Accuracy

Identifier	13
Title	Accuracy
Requirement	Our system will insure at least 80 percent accuracy of the output generation
Source	Task analysis
Rationale	Application Goal
Restrictions and Risks	None
Dependencies	Training and Testing
Priority	High

Table 3.13 shows the nonfunctional requirement of the application. The above table shows the Accuracy functionality divided into separate components for better understanding.

3.3 Use Case Diagrams

The dynamic behaviour of a system is represented by a use case diagram. Use cases, actors, and their interactions are included to encapsulate the functionality of the system. It simulates the duties, services, and operations needed by a system or application subsystem.

3.3.1 USLR Application Use Case

The main use case of this application defines the whole interaction system of our project. It specifies the flow of control and the functionalities that are associated with each use case with which the user interacts.

Figure 3.1 shows the use case diagram of our application. The figure mentions the possible roles along with their functionalities, this image represents the dynamic functionalities of the system The user can learn as well as translate the hand signs.

Figure 3.1: System - Use case diagram

3.3.2 Normal Person Use Case

The normal person use case of this application defines the whole interaction system that is subjected for the normal person. It specifies the functionalities and the flow of actions that are associated with each use case for the normal person. The normal person can view the whole application but specific defined functions that were created for the user are mentioned in the below figure.

Figure 3.2 shows the use case diagram of normal person along with their functionalities, the user can translate signs made by the disable person.

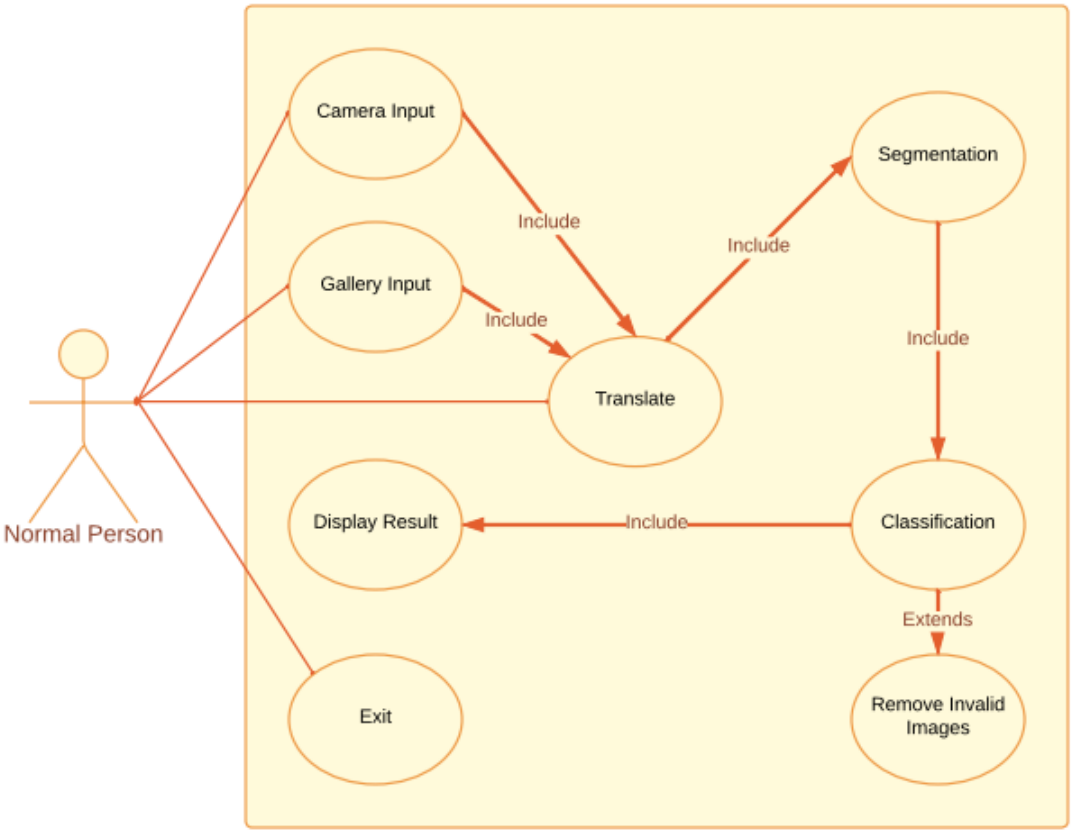

Figure 3.2: System - Use case diagram

3.3.3 Disabled Person Use Case

The same functionalities are defined for the disabled person along with the feature to learn the hand signs as well, if currently they are unaware of the sign language.

Figure 3.3 shows the use case diagram of disabled person along with their functionalities, the disable person can learn as well as translate the hand signs.

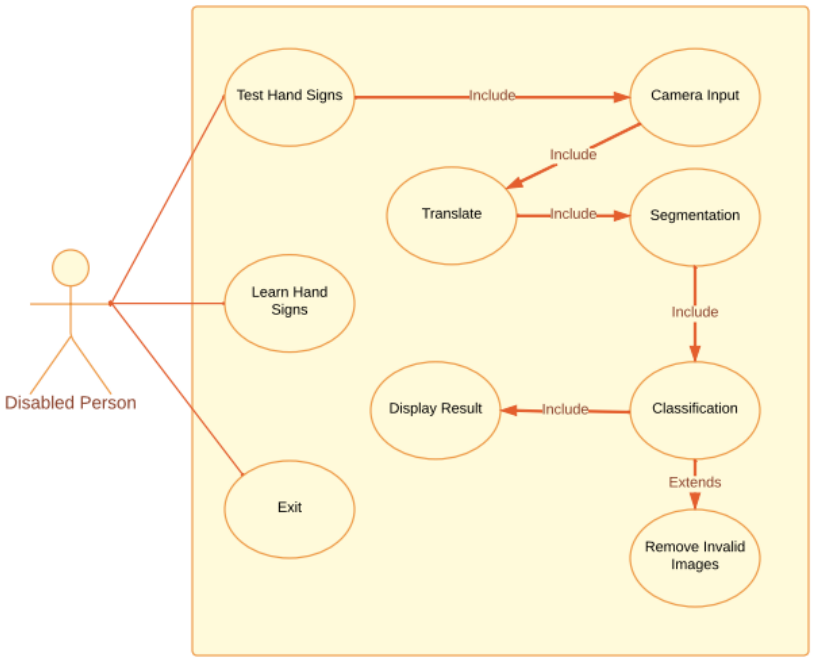

Figure 3.3: System - Use case diagram

3.3.4 Description Table - Camera Input

This use case describes that a user can take the input image through the camera, while making a hand sign and the user will capture and use that live image for further processing.

Table 3.14: Camera Input Use Case Description

Use Case ID	UC-01
Use Case Name	Camera Input
Description	Process will capture Urdu language hand signs
Primary Actor	Normal Person
Secondary Actor	Disabled person
Pre-Conditions	The hand sign should be made in front of camera
Post-Conditions	The image will be set ready to start the processing
Flow of Events	**Detail of the action**
Actor Action	User will perform the hand gesture of Sign language
System Action	System will process the image
Alternative Flow	None

Table 3.14 describes the use case of camera input. It includes all steps and constraints to be followed during the user inputs the image.

3.3.5 Description Table - Gallery Input

This use case describes that a user can provide the input image to the system, while using the image that is already taken and can be imported from the gallery.

Table 3.15: Gallery Input Use Case Description

Use Case ID	UC-02
Use Case Name	Gallery Input
Description	Image will be imported from the mobile gallery
Primary Actor	Normal Person
Secondary Actor	None
Pre-Conditions	Image must be available in the gallery
Post-Conditions	The image will be set ready to start the processing
Flow of Events	**Detail of the action**
Actor Action	Import the image form the gallery
System Action	System will process the image
Alternative Flow	None

Table 3.15 describes the use case of gallery input. It includes all steps and constraints to be followed during the user inputs the selected image through the gallery, Only the static images can be uploaded for translation.

3.3.6 Description Table - Translation

This use case describes that once the image is given for processing the system must start processing the input image in order to determine the matching Urdu character.

Table 3.16: Translate Use Case Description

Use Case ID	UC-03
Use Case Name	Translation
Description	Translation process will start
Primary Actor	Both
Secondary Actor	None
Pre-Conditions	The image is imported or captured for processing
Post-Conditions	The image is transferred for further processing
Flow of Events	**Detail of the action**
Actor Action	Press the translation button to start the process
System Action	System will start processing the image
Alternative Flow	None

Table 3.16 describes the use case of translation. It is the main button that basically initiates the translation process. The user should upload an image and the press the translate button to start the processing.

3.3.7 Description Table - Segmentation

This use case describes that processing for the segmentation of the hand from the image. Basically the system will extract the hand from the image even if the any other person is also present in the image.

Table 3.17: Segmentation Use Case Description

Use Case ID	UC-04
Use Case Name	Segmentation
Description	This process will segment out the images' AOI
Primary Actor	Both
Secondary Actor	None
Pre-Conditions	The image is in the system for processing
Post-Conditions	The segmented image is sent further for classification
Flow of Events	**Detail of the action**
Actor Action	The image is captured
System Action	System starts processing the captured image
Alternative Flow	None

Table 3.17 describes the use case of segmentation. It's one of the main functionalities of the system where the system will segment out the hand from the image.

3.3.8 Description Table - Classification

This use case describes that once the AOI is extracted from the input image then the system will perform some pre-processing techniques on that extracted image and further that image will be used for the classification purpose.

Table 3.18: Classification Use Case Description

Use Case ID	UC-05
Use Case Name	Classification
Description	Input image will be classified as matched or not
Primary Actor	Both
Secondary Actor	None
Pre-Conditions	The image must be segmented
Post-Conditions	The image will be classified as matched or not
Flow of Events	**Detail of the action**
Actor Action	Capture or import the image
System Action	Process the image and classify as matched or not
Alternative Flow	None

Table 3.18 describes the use case of classification. It is also one of the main functionalities of the system where the system will successfully segmented hand should be correctly classified if it is a character that matches with data-set or not.

3.3.9 Description Table - Remove Invalid Images

This use case describes that once the image is classified as the sign the result will be returned to the user and the image will be stored in the database to further improve the data set. But if the image is invalid the system will discard that image.

Table 3.19: Remove Invalid Images Use Case Description

Use Case ID	UC-06
Use Case Name	Remove Invalid Images
Description	This process will discard the invalid image taken
Primary Actor	Both
Secondary Actor	None
Pre-Conditions	Image must be captured
Post-Conditions	Image will be deleted
Flow of Events	**Detail of the action**
Actor Action	Capture the hand sign
System Action	System will remove the image
Alternative Flow	None

Table 3.19 describes the use case of remove invalid image. In this stage if an image is classified as invalid, then system will display the message of invalid image.

3.3.10 Description Table - Display Result

This use case describes that once the image is classified as the sign the result will be returned to the user then the resultant Urdu alphabet will be displayed on the users screen.

Table 3.20: Display Result Use Case Description

Use Case ID	UC-08
Use Case Name	Display Result
Description	Result of the image will be displayed
Primary Actor	Both
Secondary Actor	None
Pre-Conditions	The image must be classified as a hand sign
Post-Conditions	The alphabet will be displayed on the screen
Flow of Events	**Detail of the action**
Actor Action	The user will capture the image
System Action	The system will display the matched image result in form of Urdu language alphabet
Alternative Flow	None

Table 3.20 describes the use case of display result. It is explained that how user will give input to the system and how system will generate output against user's input.

3.3.11 Description Table - Exit

This use case describes that once user has fulfilled his/her need by using this application, and upon the exit the system should ask them for final in case if they might need to use this application again.

Table 3.21: Exit Use Case Description

Use Case ID	UC-09
Use Case Name	Exit
Description	The button will shutdown the application
Primary Actor	Normal
Secondary Actor	None
Pre-Conditions	User must be in main menu
Post-Conditions	The application will shutdown
Flow of Events	**Detail of the action**
Actor Action	The user will press the exit button
System Action	The system will shutdown the application
Alternative Flow	None

Table 3.21 describes the use case of exit. It is explained that once the user is done using the application, after that user can simply logout from the application.

3.3.12 Description Table - Test Hand Sign

This use case describes the functionalities that once user has learnt the hand signs they can test either that they have learned them correctly or not. The system will identify the signs as matched or not matched.

Table 3.22: Test Hand Sign Use Case Description

Use Case ID	UC-10
Use Case Name	Test Hand Sign
Description	Person will be able to test, if signs learnt are accurate
Primary Actor	Normal Person
Secondary Actor	None
Pre-Conditions	The user must be in the Lean mode
Post-Conditions	The system will show the matched sign result
Flow of Events	**Detail of the action**
Actor Action	The User will capture his/her hand sign image
System Action	System will start processing the captured image
Alternative Flow	None

Table 3.22 describes the use case of test hand sign. That once user has learnt any particular hand sign, then he/she can practice them through translate option.

3.3.13 Description Table - Learn Hand Sign

This use case describes the functionalities that if user does not know the sign language they can learn it using the option available in the menu.

Table 3.23: Learn Hand Sign Use Case

Use Case ID	UC-11
Use Case Name	Learn Hand Sign
Description	Use Learn mode to start learning the hand signs
Primary Actor	Learner
Secondary Actor	None
Pre-Conditions	User should be in the main menu of the application.
Post-Conditions	Learn the alphabet by using alphabet buttons
Flow of Events	**Detail of the action**
Actor Action	Click on the Alphabet to learn, Skip learned Alphabets
System Action	Signs will be displayed
Alternative Flow	None

Table 3.23 describes the use case of learn hand sign. If any user wants to learn hand signs as well, then the user can learn the hand signs through the learn option.

Chapter 4

Design

The process of defining a system's modules, interfaces, components, and data in order to meet predetermined requirements is known as system design. The process of building or changing a system, as well as the procedures, techniques, models, and development approaches, is referred to as system development.

4.1 System Architecture

When we built the deep Urdu Sign Language Recognition System, we went with a blackboard-style architecture and a procedural method.

The reason is that our system involves different stages, such as image processing, sign detection, sign segmentation, and sign classification, which can be thought of as knowledge sources for the blackboard. The blackboard architecture stores the intermediate processing results, detection results, segmentation results, and classification results as recognition hypotheses in the blackboard data structure. The learning sources contain all the algorithms required for optical character recognition. It's up to the control parts to choose an item on the blackboard and a source of information for that item to be used.

4.2 High Level Design

In addition to describing the overall architecture and description of any application, it alludes to the system's overall design. System/macro-level design is another name for it. The High-Level Design (HLD) diagram will explain how the system will develop.

4.2.1 Architecture Diagram

A visual depiction that depicts the actual physical implementation of a software system's components is called an architecture diagram. It displays the relationships, constraints, and boundaries between each piece as well as the overall structure of our project.

Figure 4.1 shows the system architecture. It is a graphical representation that depicts the physical implementation of our application.

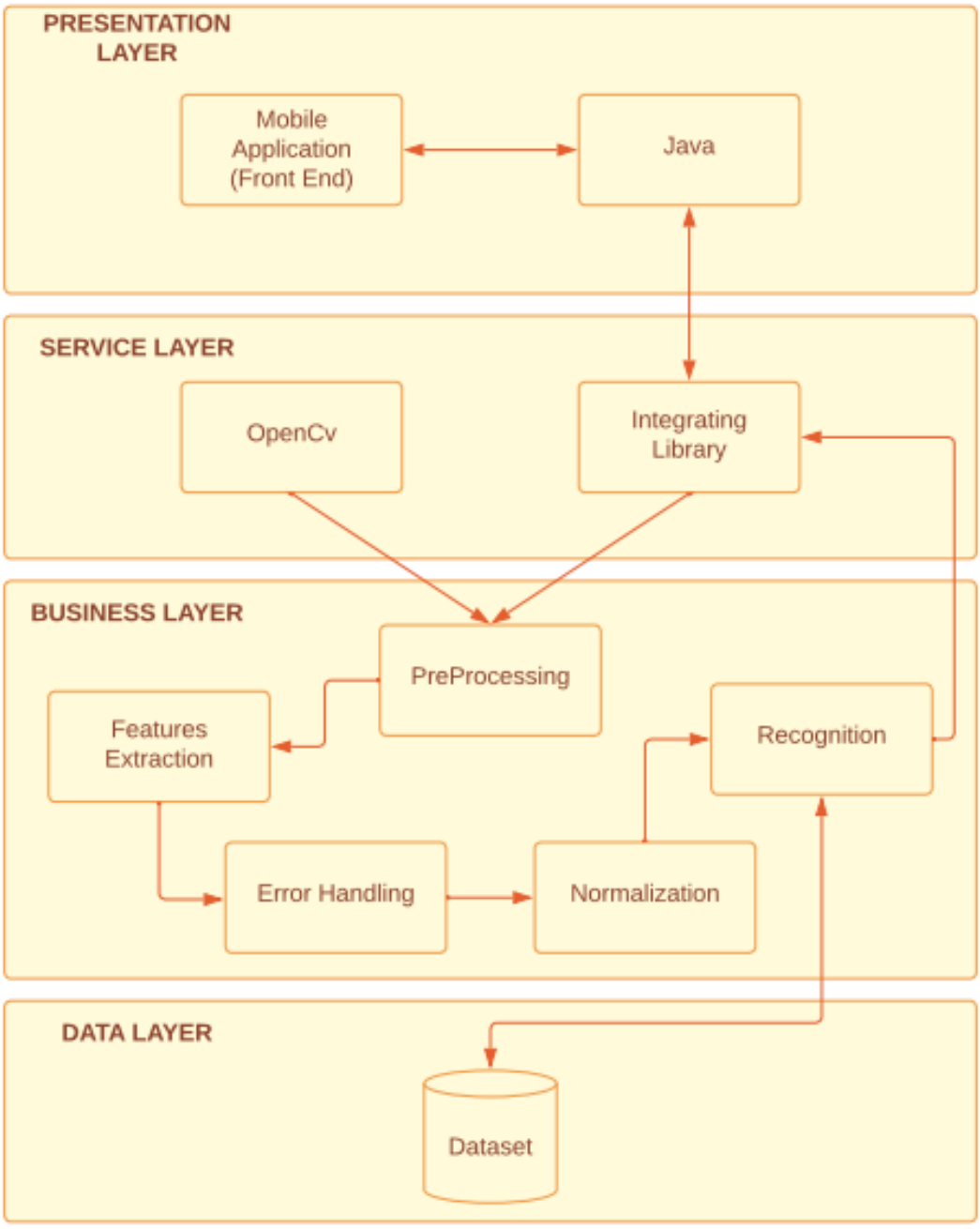

Figure 4.1: System Architecture - USLR

4.2.2 Activity Diagram - Camera Input

The activity diagram shows the software processes as a progression of actions. In this activity diagram the steps for Camera Input have been defined. These actions are carried out by the user, while it also describe business processes and use cases that rely under the flow of this activity.

Figure 4.2 shows the activity diagram, when the user initiates the process of translation while using the live picture.

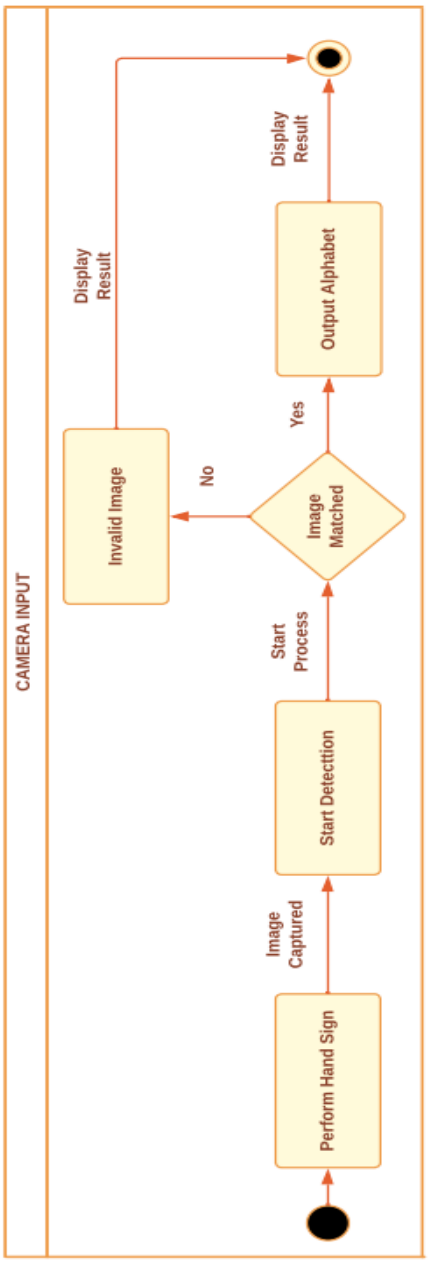

Figure 4.2: Activity Diagram for Camera Input

4.2.3 Activity Diagram - Gallery Input

In this activity diagram the steps for Gallery Input have been defined. These actions are carried out by the user in order to give input to system using the image already present in device gallery, while it also describe business processes and use cases that rely under the flow of this activity.

Figure 4.3 shows the activity diagram, when the user initiates the process of translation while using the selected picture from gallery.

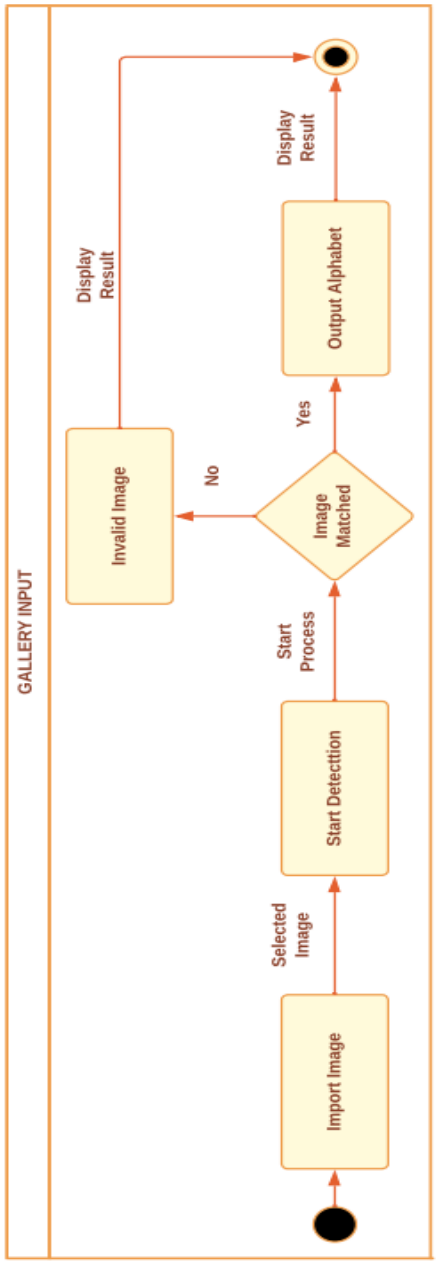

Figure 4.3: Activity Diagram for Gallery Input

4.2.4 Activity Diagram - Translation

In this activity diagram the steps for Translation of the hand sign image into Urdu alphabet have been defined. These actions are carried out by the user in order to translate the hand sign by initiating this process using the translate button present on screen, this activity diagram also describe business processes and use cases that rely under the flow of this activity.

Figure 4.4 shows the activity diagram, when the image classification process starts. It shows the flow of information for the back end procedure.

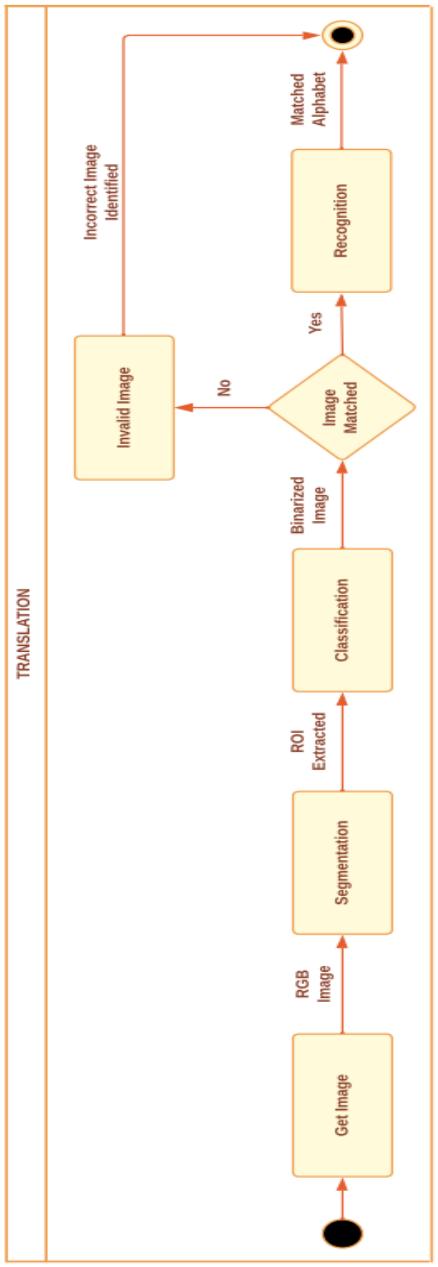

Figure 4.4: Activity Diagram for Translation

4.2.5 Activity Diagram - Test Hand Sign

In this activity diagram the steps for Testing the hand sign once the user has learned them. The user can simple follow the translation step after learning the signs from the sign menu defined in the application.

Figure 4.5 shows the activity diagram for testing hand signs. This activity contains the whole procedure that is followed while the process of translation when the user has learnt and wants to practice the signs he/she learnt.

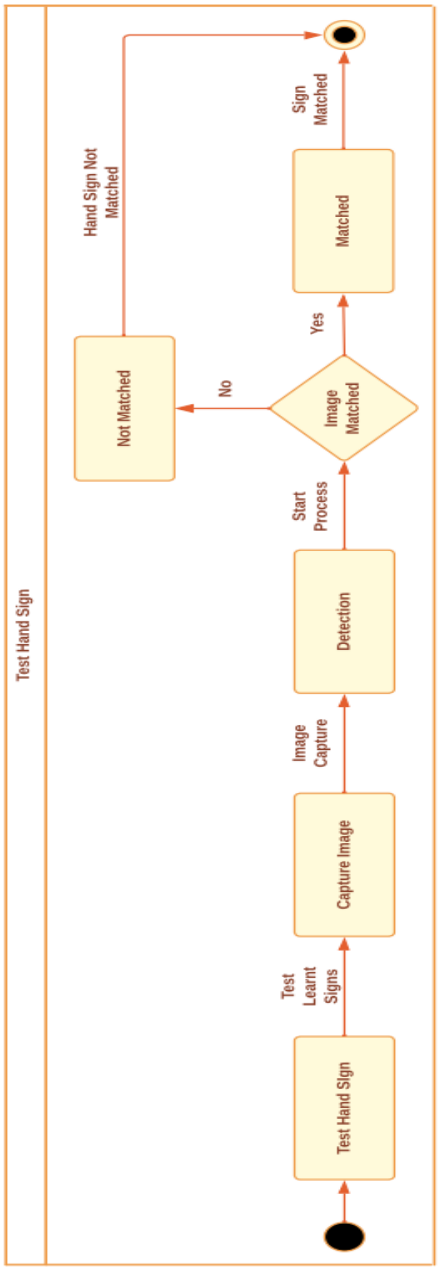

Figure 4.5: Activity Diagram for Test Hand Sign

4.2.6 Activity Diagram - Learn Hand Sign

In this activity diagram the steps for Learning the hand signs. The ease for user has been provided in the application along with user manual through which if the non-signer is interested in learning the hand sign, then he/she can learn it without having the need to hire an instructor.

Figure 4.6 shows the activity diagram for learning hand signs. This activity contains the complete list of all the characters and their corresponding hand signs.

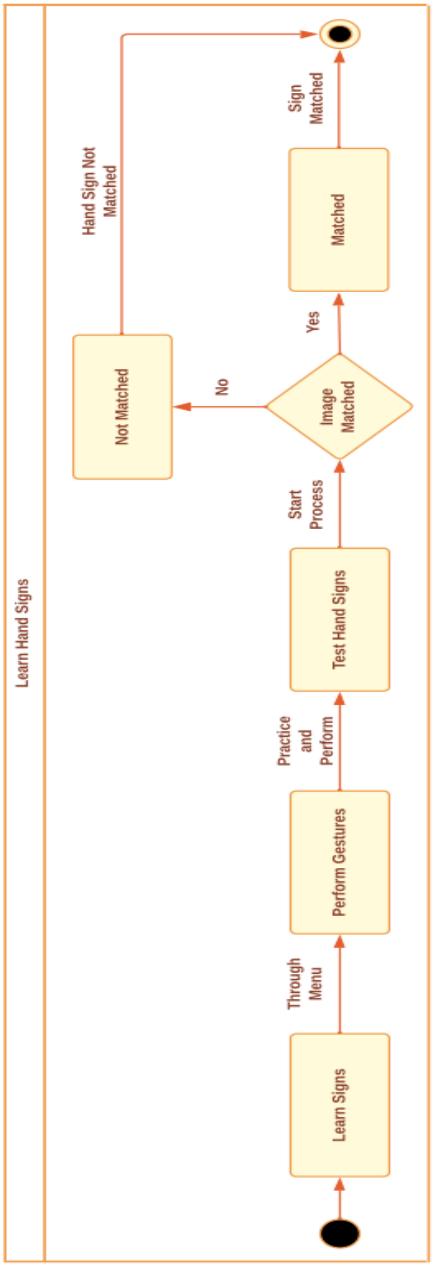

Figure 4.6: Activity Diagram for Learn Hand Sign

4.3 Low Level Design

Every software module is specifically described in the low-level design. By including the reasoning behind each system component, it provides a thorough description of every module. It provides a micro-level design and goes deeply into each system's specification. This Low-Level Design (LLD) is basically a component-level design method that moves forward in a step-by-step way.

4.3.1 Data Flow Diagram - DFD (Level 0)

Context Diagrams are another name for DFD Level 0. This diagram basically provides a general summary of our entire system that has been modelled. The system is being displayed as a single, high-level process, together with its relationship to its users, while providing an overview view.

Figure 4.7 shows the zero level data flow diagram in which basic overview of the whole system is being analyzed.

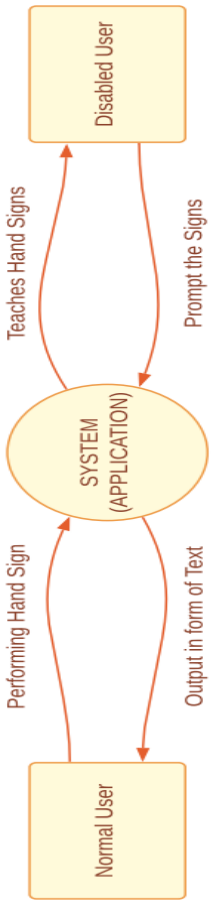

Figure 4.7: Data Flow Diagram (Level 0) – USLR

4.3.2 Data Flow Diagram - DFD (Level 1)

In this diagram the Context Level Diagram is broken down more specifically in DFD Level 1. This diagram basically gives an in detail overview of our project as it deconstruct the context diagram's high-level process into its constituent parts.

Figure 4.8 shows the level one data flow diagram in which the translation process is being broken down into sub processes.

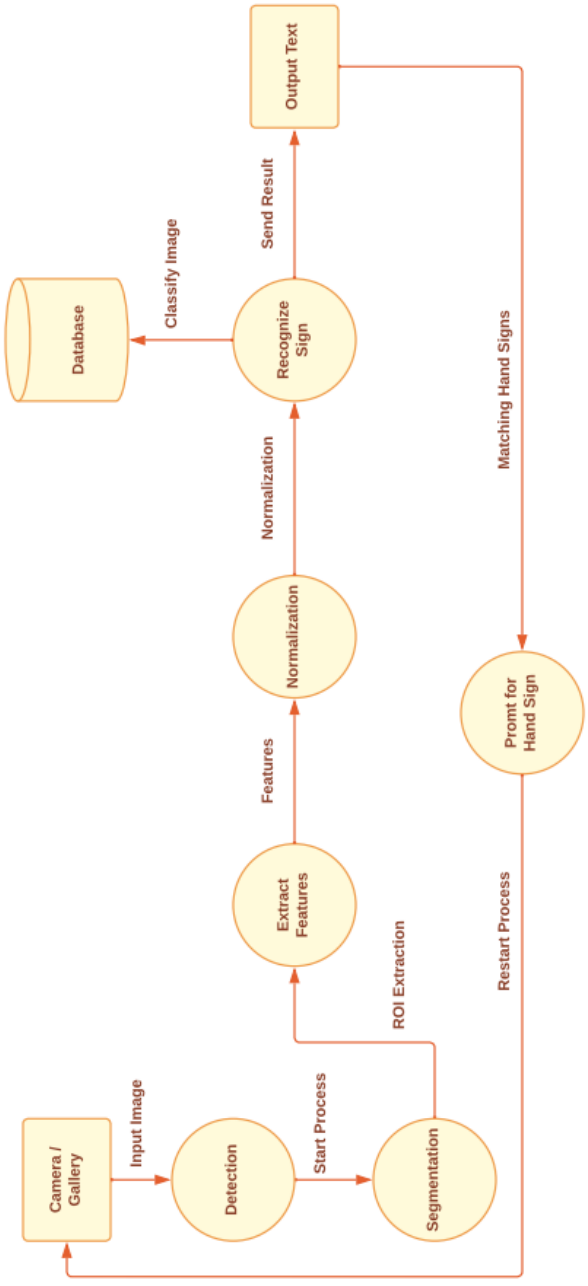

Figure 4.8: Data Flow Diagram (Level 1) – USLR

4.3.3 Sequence Diagram - Camera Input

This diagram basically illustrates the interactions between the user and the application while capturing the input image using camera, and also defines the order in which they occur, a sequence diagram is a form of interaction diagram. It clearly shows the interaction of user with each module of the application and at which sequence they occur while taking the input image through camera.

Figure 4.9 shows the sequence diagram for the normal person, it shows the process interactions arranged in time sequence when the user gives the input through camera.

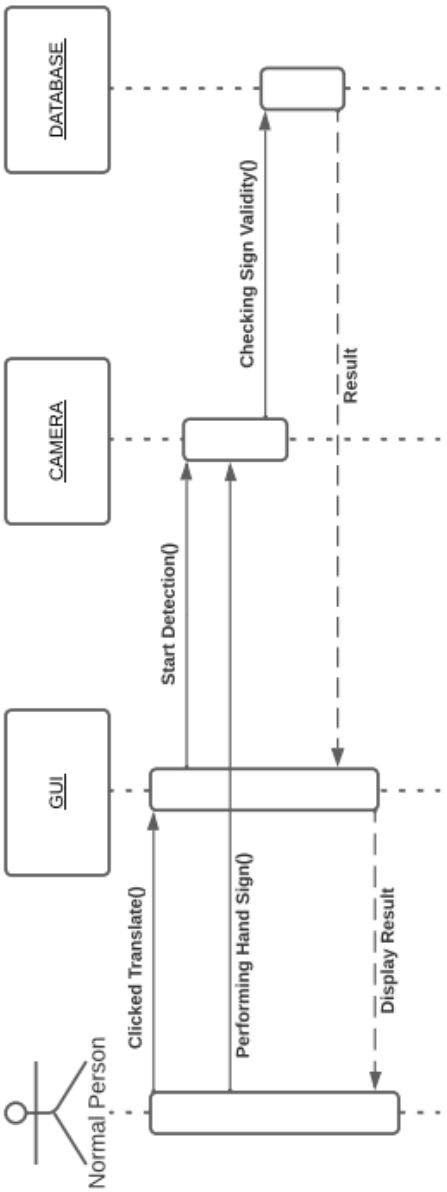

Figure 4.9: Sequence Diagram for Camera Input

4.3.4 Sequence Diagram - Gallery Input

This diagram basically illustrates the interactions between the user and the application module for using the application in order to import image from gallery to use it for further processing. It defines the interaction of user with module of the application for importing the image and at which sequence they occur.

Figure 4.10 shows the sequence diagram for the normal person, it shows the process interactions arranged in time sequence for the process that undergoes when the user gives the input through gallery.

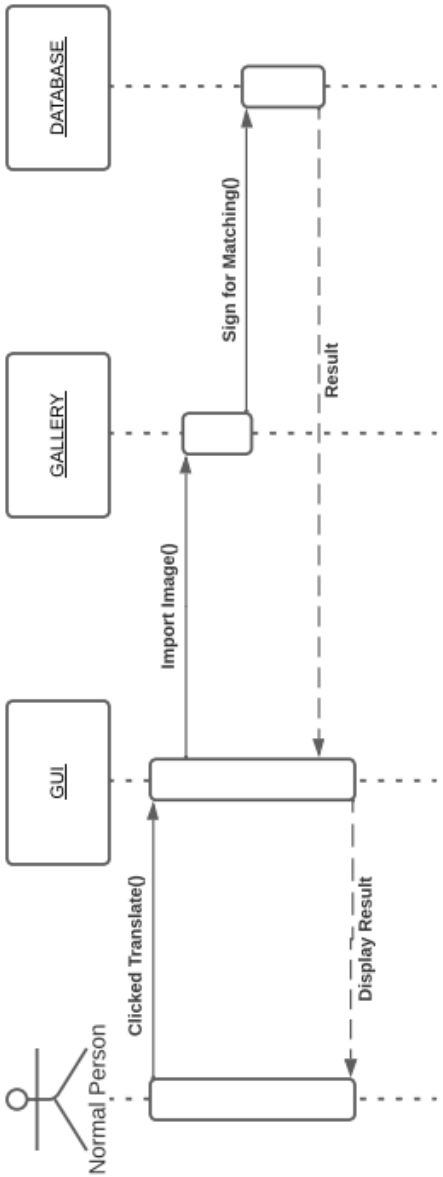

Figure 4.10: Sequence Diagram for Gallery Input

4.3.5 Sequence Diagram - Translation

This sequence diagram defines the whole interaction of the user for the translation process, the order in which they occur in the system.

Figure 4.11 shows the sequence diagram for the normal person, while it shows the process interactions arranged in time sequence for the whole process that took place in translating the hand sign image.

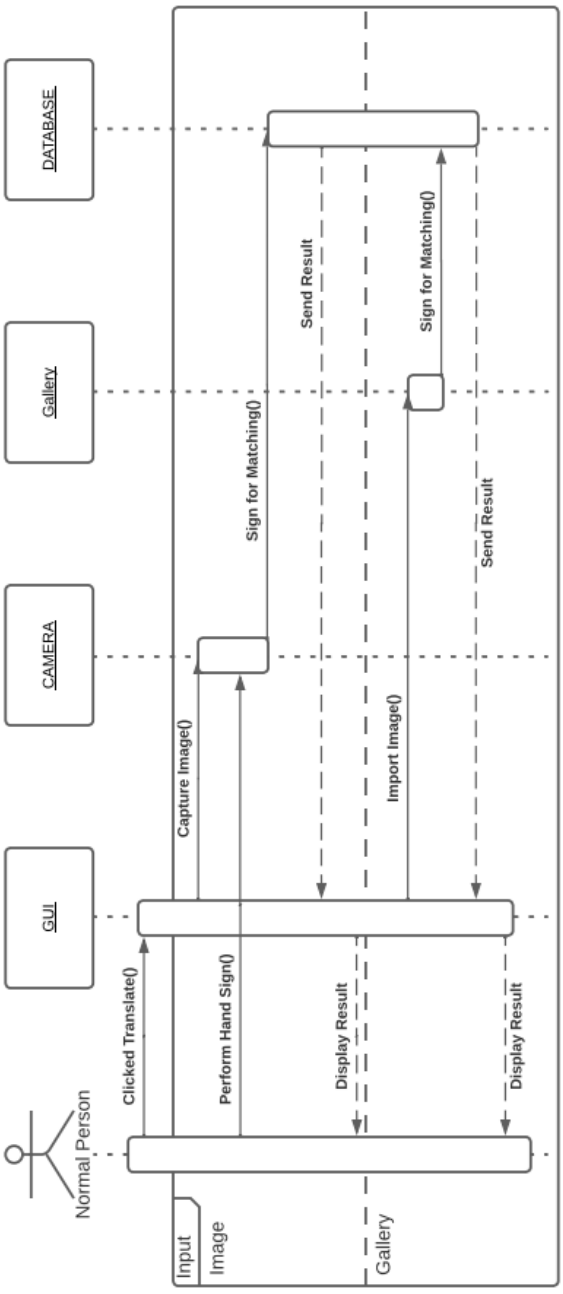

Figure 4.11: Sequence Diagram for Translation

4.3.6 Sequence Diagram - Learn Hand Sign

This sequence diagram defines the whole learning process of the user to learn the hand signs. It also states the sequence of actions that occur in the system for this process of learning.

Figure 4.12 shows the sequence diagram for the disabled person, in which it shows the process interactions arranged in time sequence for the whole process through which the disabled person can learn and check for the results if he/she has learned them correctly.

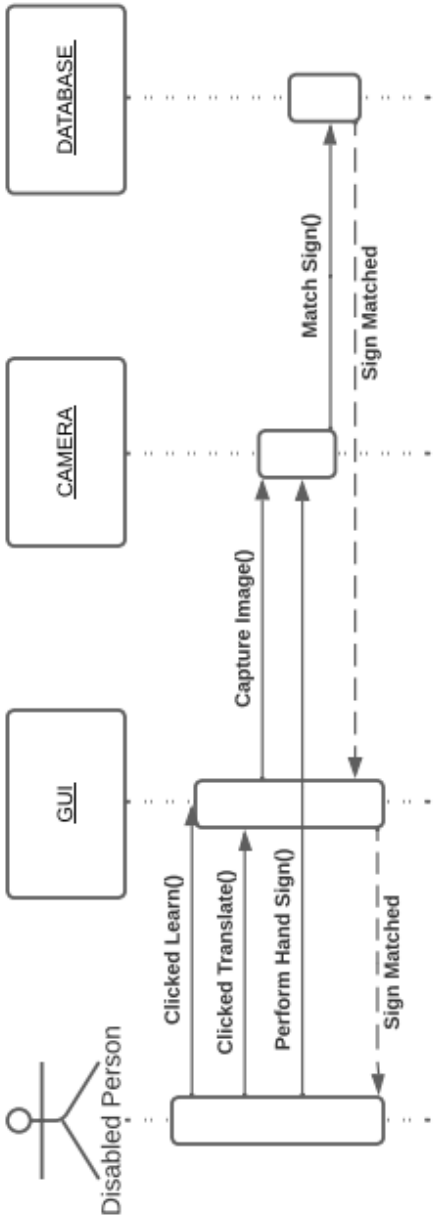

Figure 4.12: Sequence Diagram for Learn Hand Sign

4.4 State Diagram

An depiction of the states an object in the UML can reach as well as the transitions between those states is called a state diagram, also known as a state machine diagram or state chart diagram.

Figure 4.13 is the state diagram for our system, it is representation of the states an object can attain during processing as well as the transitions between those states in the UML.

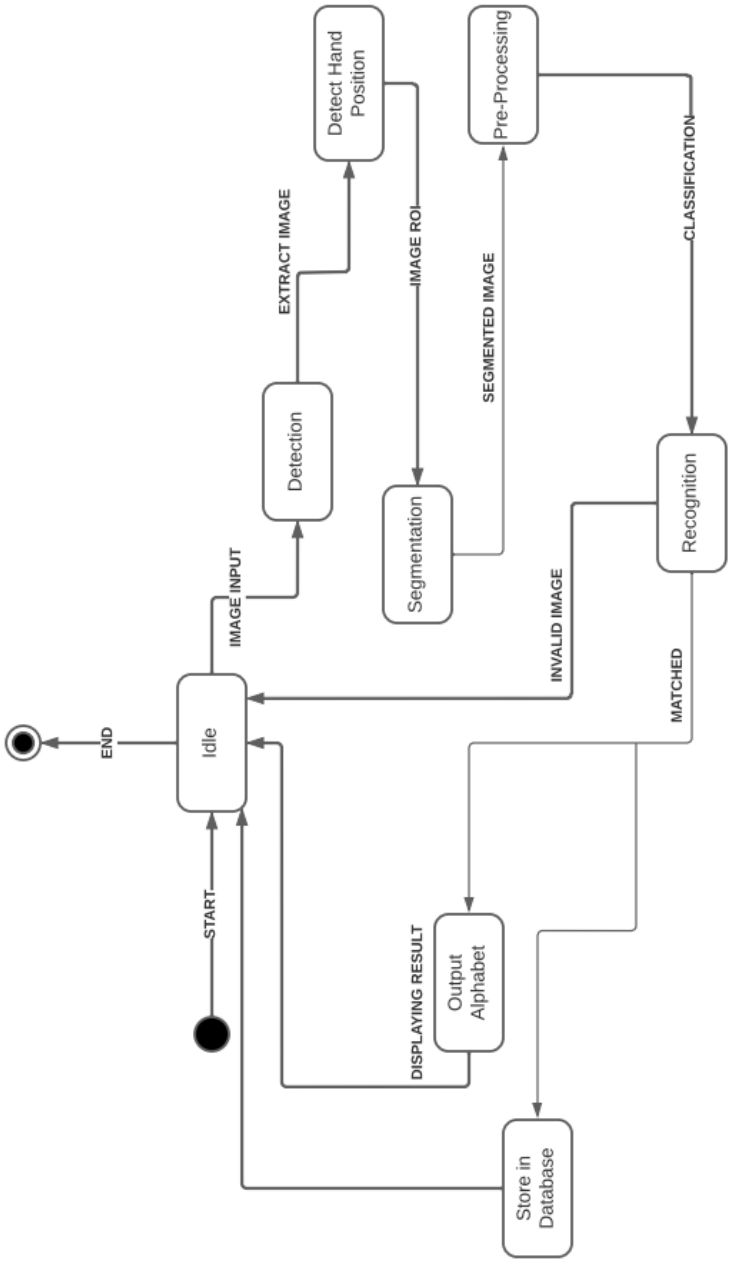

Figure 4.13: State Diagram for USLR

4.5 Communication Diagram

It serves as an example of the connections and interactions between software items in the UML. This diagram serves as a representation of the dynamic behaviour of the specific use case and clarifies each object's function within our project.

Figure 4.14 is the communication diagram for our system, in which it gives the representation of the relationships and interactions among the object in our application.

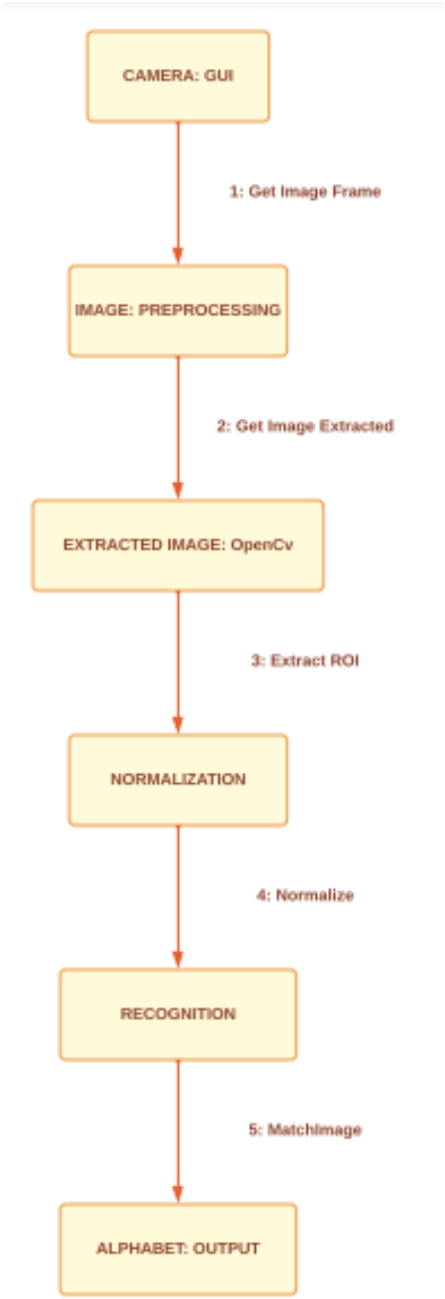

Figure 4.14: Communication Diagram for USLR

4.6 GUI Design

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user. This illustrate the User Interface design of our Mobile Application. Our application consists of several screen and each screen has their own respective functionality.

Figure 4.15 shows the splash screen of the application.The splash screen is designed to give the general overview of the application.

Figure 4.15: Splash Screen of USLR Application

4.6.1 GUI Design - Main Menu

This is the main menu screen of our application. In this current screen the user has the option to either translate the hand sign by opening the translate option or can learn new hand signs by choosing the learn option. The manual for the usage of the application is also provided in the Information button at the bottom right option.

Figure 4.16 shows home page. After splash screen the user will be directed to home page. In home page, user will select translate or learn option as per the requirements. Once selected, the required page will be loaded.

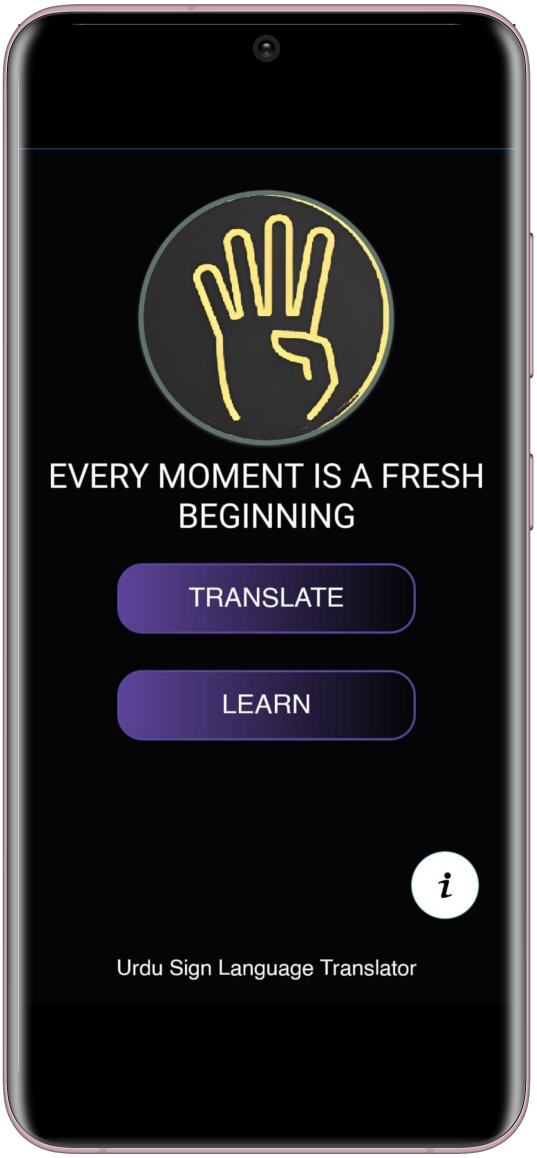

Figure 4.16: Main Menu of USLR Application

4.6.2 GUI Design - User Manual

This screen provides the user manual to the user so that he/she can learn how to properly use this application and how can they get the maximum output from this application in term of learning as well as in terms of translation. This user guide provide complete information from start to end.

Figure 4.17 shows user manual. The user can learn about the general features about the application and can understand how to use it.

Figure 4.17: User Manual of USLR Application

4.6.3 GUI Design - Translation Mode

This is the main hub of our project. This is the screen where the translation of the hand sign images are being is carried out. The user has to select the input method of the image either using camera or import from gallery, and then the processing begins when the translate button is pushed.

Figure 4.18 shows translate page. The user can translate the hand signs made, by selecting and proceeding with the translation process.

Figure 4.18: Translation Mode of USLR Application

4.6.4 GUI Design - Gallery Import

In this screen the processing of the image is shown that is imported from the gallery is being shown. Once the translation process begins the user is entertained with the dialog box that shows the system processing time that it will take to classify hand sign.

Figure 4.19 shows the translation process wait dialogue. The user gets informed when the processing starts and how much time will it take.

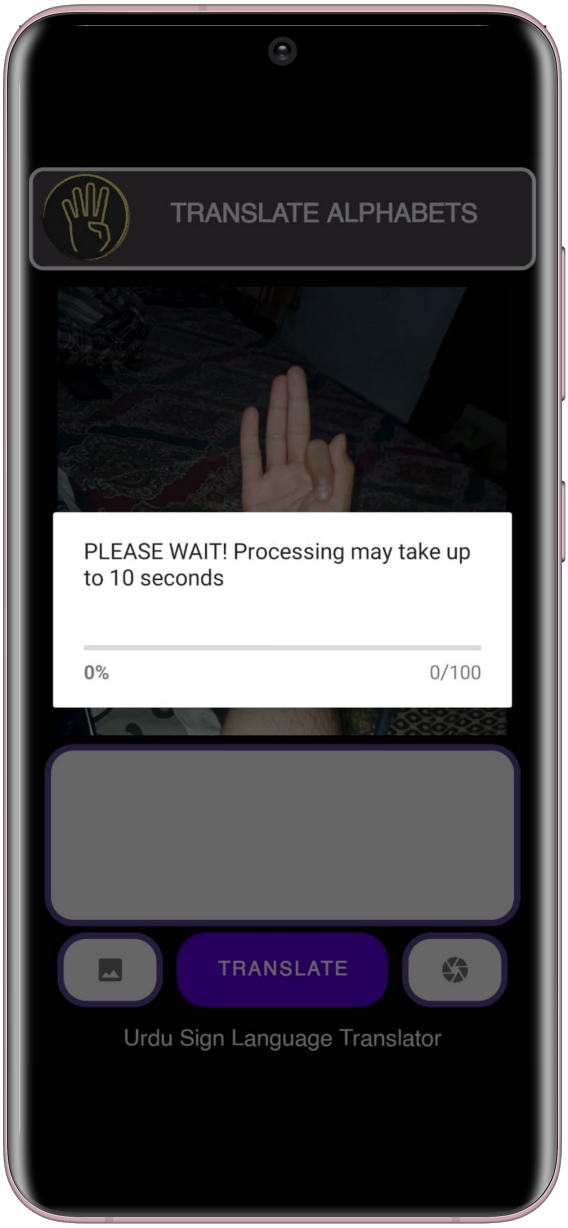

Figure 4.19: Import Image from Gallery Option of USLR Application

4.6.5 GUI Design - Processed Image

This screen shows the returned result after the processing of the image that is given as an input either by image imported from gallery or by using the image that is taken from the camera of the of the application.

Figure 4.20 shows the screen with returned result. As show in the figure the system has correctly identified the image and the result is being displayed on the translate screen.

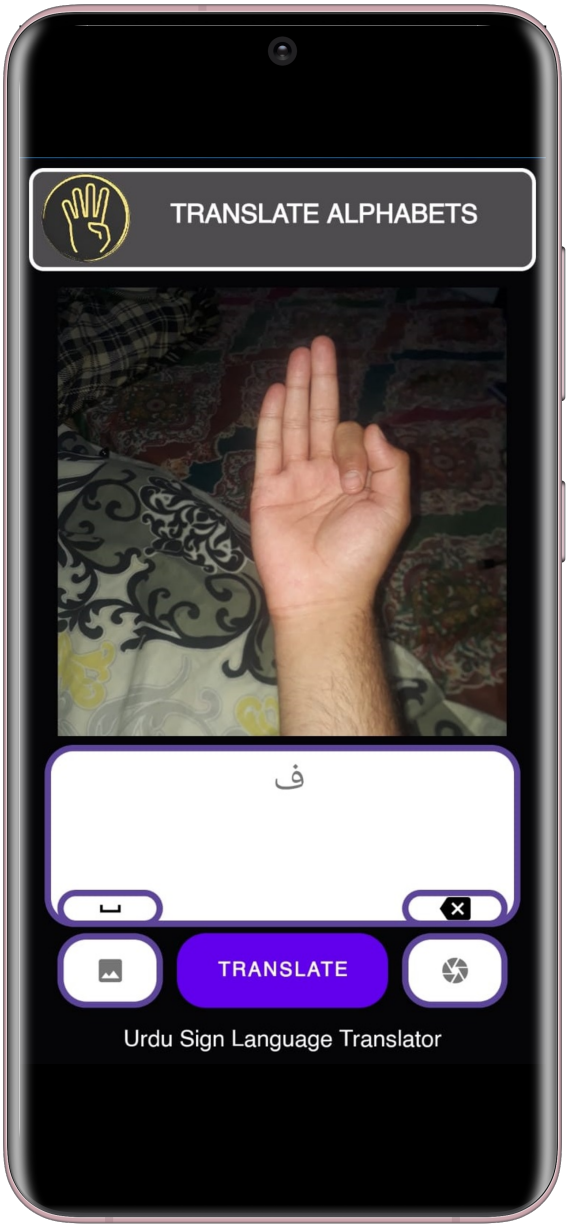

Figure 4.20: Returned Result of image in USLR Application

4.6.6 GUI Design - Camera Input

In this screen the processing of the image is shown that is taken as input using the camera intent in the translation screen. The user can capture the hand sign and then use it for the processing as well to classify that captured hand sign.

Figure 4.21 shows the screen with returned result with image taken from the camera. The results show in this figure is from the live image taken.

Figure 4.21: Import Image from Camera Option of USLR Application

4.6.7 GUI Design - Learn Mode

In this screen the person has the ability to learn the hand signs. The user can learn the signs as per their availability. The user can learn any sign by just clicking the button that is displaying the Urdu alphabets. The alphabets make it easier for the user to open the desired sign image.

Figure 4.22 shows the learning screen. The ultimate guide to learn the hand sign. Learn screen is provided with image representing the each character with hand signs.

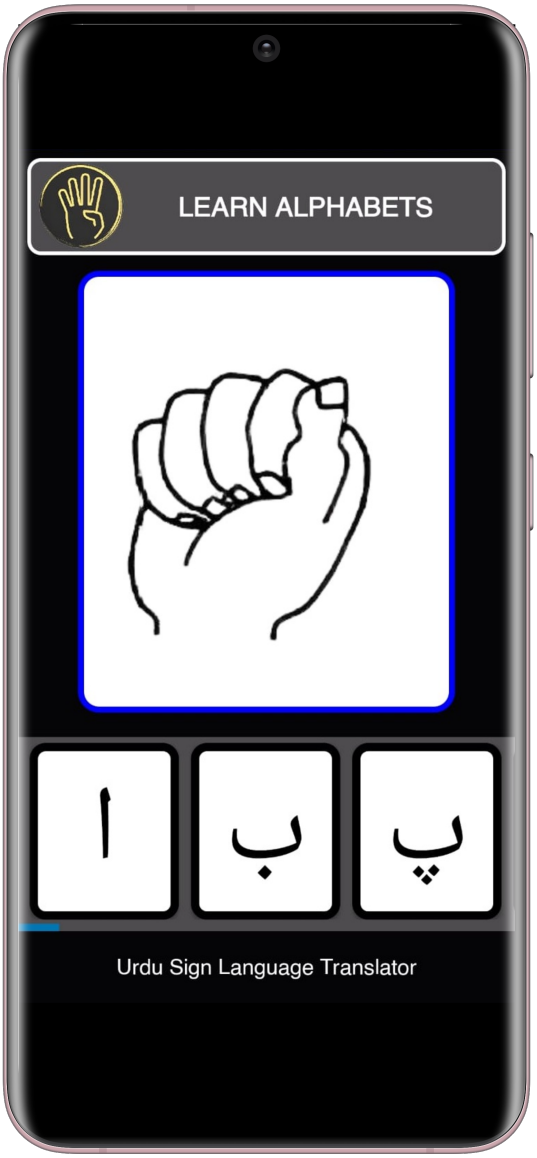

Figure 4.22: Learning Signs from Pictures of USLR Application

4.7 External Interfaces

- The mobile must have a working camera in order to clearly capture the hand sign.
- The system that is being used to build this project must have 8GB+ RAM & 256GB Hard Disk or 128GB SSD Hard Drive for smooth working experience
- The internet connection must be good for research purposes.
- There must be significant space for the data set training.

Chapter 5

System Implementation

Implementation is the process of moving an idea from concept to reality. The System implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment. This chapter includes information regarding the tools and technologies that are used in our project.

5.1 Development Environment (IDE)

Android Studio is Google's authorised IDE for developing the Android OS. It is based on the IntelliJ IDEA software made by JetBrains. It is designed for people who want to make apps for Android.

Android Studio is used to create our project's mobile application. This application can be installed on Android phones and used while connected to the internet.

5.2 Architecture and Component Integration

The system architecture is the model that describes the proposed system's structure, behavior, and various views. It also describes the system's internal components and the functionalities of the various system components. The proposed system is implemented using a variety of tools and technologies. In our project, the integration is between the Python project that serves as the back-end and the Java project with which we have created the mobile application (front-end).

5.3 System Implementation

This section includes the implementation of our project USLR, in which the Hand Sign Recognition application was developed. For the front-end of the application (UI), which is designed in Java, Android Studio (IDE) is used, and for back-end processing, we have used Python for processing and FLASK for the integration between these two modules. We have also used POSTMAN for the testing of our API.

Figure 5.1 shows the interface of the android studio the IDE which was used to create the android application using the Java language.

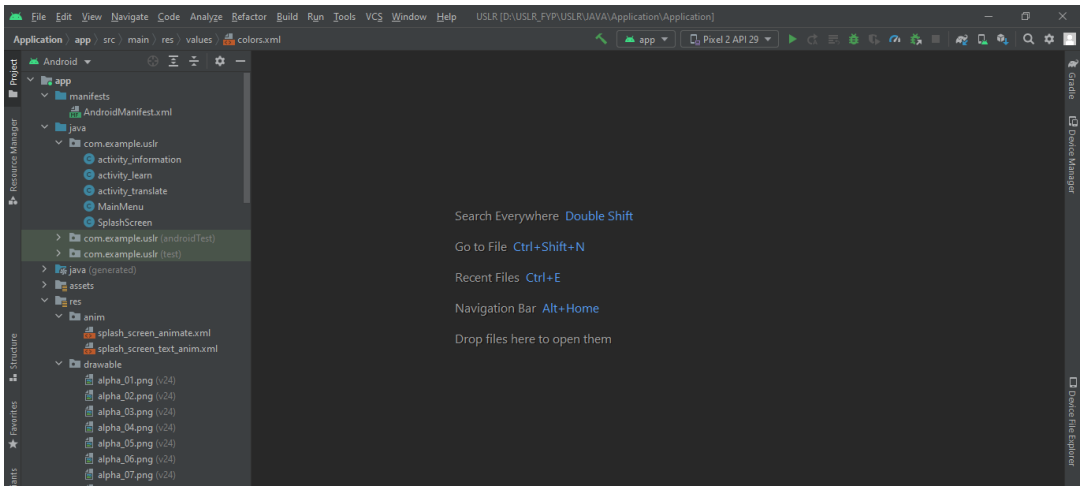

Figure 5.1: Android Studio Interface

Python is a high-level, interpreted, general-purpose programming language. With the usage of extensive indentation, its design philosophy places an emphasis on code readability. Python uses garbage collection and dynamic typing. Python is also used for deep learning and machine learning project, and as our project is using deep learning concepts that is why we chose to work with python.

Figure 5.2 shows the interface of the spyder another IDE which was used to create the back end of the application using the Python language.

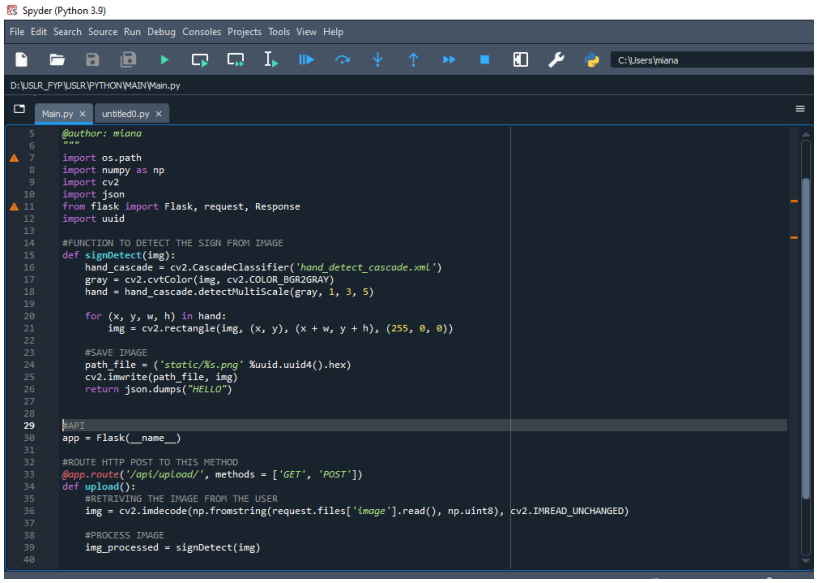

Figure 5.2: Spyder Interface

5.4 Backend Implementation

This is the original image that the user gives as an input to the system. The system receives this image and then start working on this image to predict the Urdu letter.

Figure 5.3 shows the image that was received at the back end for processing. The image is a RGB channel that will be used later for further processing.

Figure 5.3: Original Image Given As Input

This is the segmented image that will be compared with the original image and after that using bit-wise operator the system will extract the usable pixels from the real image and then that image will be used for further processing.

Figure 5.4 shows the segmentation process from the image in which the system has extracted the skin from the image based of color based segmentation.

Figure 5.4: Extracted Hand from Input Image

This is the image that basically show the contours, and from these contours we will use the longest contours chain and to get the ROI and after that the dimension of that hand from these contours will be extracted.

Figure 5.5 shows the ROI extraction process from the segmented image in which the system has extracted the hand from the image while using the contours.

Figure 5.5: Finding Contours of Extracted Image

This is the extracted hand image that will further be used for the processing and the later on the classification will be carried out using this image as a base.

Figure 5.6 shows the image after the ROI extraction process, this extracted image will be used for comparison and the desired pixels will be separated and binarized.

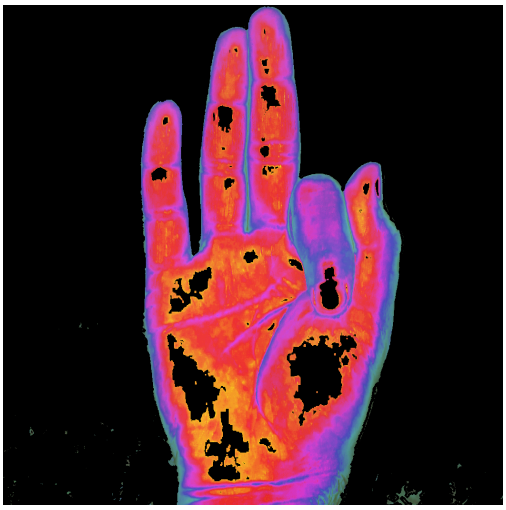

Figure 5.6: Extracted ROI from Image

This is the final image the binary image that will be used for the classification purpose to predict from that model that which hand sign does it represents.

Figure 5.7 shows the binary image that will be used for comparison and to classify if the sign made was matching or not.

Figure 5.7: Pre-processed Image used for Classification

5.5 Application Structure

The applications architecture describes the behavior of our application that how it is being used, what functionalities it is focused on and how the user is interacting with our application.

Figure 5.8 shows the structure of the application in which all the objects representing the application are being observed.

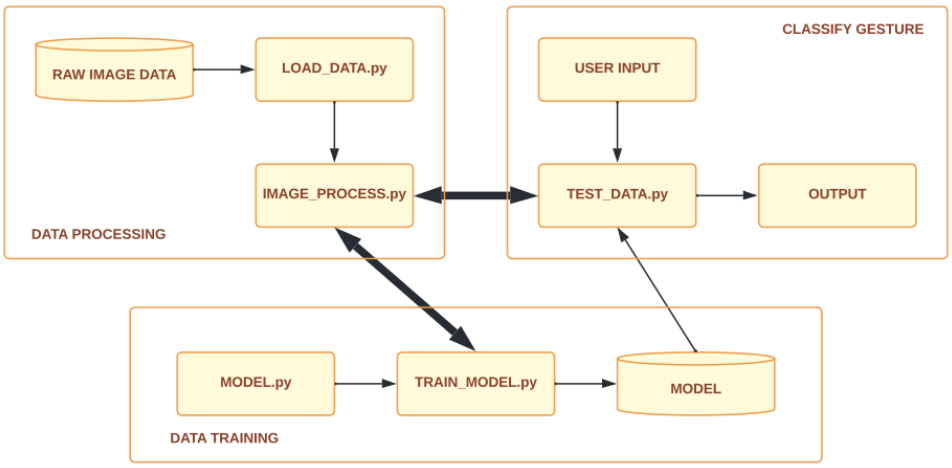

Figure 5.8: Application Structure of USLR

5.6 Tools and Technologies

The applications were created in Java and Python on the Android Studio (IDE) and Spyder (IDE). We use OpenCV to analyse images captured by a mobile phone's camera to detect hand gestures. These are the tools and technologies used in the development of our project.

- Computer
- Android Phone (v 5.0)
- Android Studio (v 2021.1.1)
- Spyder (v 5.1.5)
- Jupyter Notebook (v 6.4.5)
- Java (v 11.0.11)
- Python (v 3.9)

5.7 Preprocessing and Initialization

Complete process that is being carried out in the backend of our system. The whole process is listed below in order of their execution sequence.

- Start
- Input given as RGB image
- Down sampling the image
- Converting RGB image to Gray scale
- Converting Gray scale image to Binary image
- ROI segmentation
- Normalizing image
- Identifying the boundary of hand sign
- Classifying the hand sign
- Returning Result

5.8 Libraries

We are using two separate IDEs for the development of the front-end and back-end. We will have different modules and different libraries that are used in our project, some of which are mentioned below:

- TensorFlow

  TensorFlow is a free and open-source ML and AI-based software library. This module was utilized for CNN neural network training and hypothesis.

- Keras

  Keras is a Python interface for neural networks that is open-source software. It serves as a gateway to the TensorFlow Python module.

- OpenCV

  OpenCV is a module in Python that is primarily geared towards real-time computer vision. This library is used to process images.

- Flask

  Flask is a micro web framework that is built on the Python programming language.. It is called a microframework since it does not require the use of any particular tools or libraries. The API was created using this library.

- NumPy

  NumPy is a Python module that provides support for massive, multi-dimensional arrays and matrices, as well as a large variety of high-level mathematical operations for manipulating them. This library was used to work with huge matrices of pixel-format image data.

- Universally unique identifier (UUID)

  A Universally unique identifier (UUID) is a 128-bit label used in computer systems to store information. This library was used to produce the id for the generated server request.

5.9 Processing Logic

Processing logic for our project will be followed as listed below in the following steps:

- Application Start
- Capture Hand Sign or Import from Gallery
- Image Processing
- Identify the target gesture
- Image Recognition

- Image Classification
- Returning the results

Figure 5.9 shows the graphical representation of the processing logic used in the sign classification. The overall concept of the modules that are being used are mentioned.

Figure 5.9: Processing Logic of Gesture to Text

5.10 Trained Data Accuracy

Multiple copies from the data-sets available on Keggle were used for the creation of our data set, totaling 37,000 images, of which 29,600 are used for training and 7400 are used for testing. The images used in the model training and testing were converted to binary, The final accuracy obtained was sufficient to justify continuing with this model. The CNN model was used to train the data set for our project.

Figure 5.10 shows the trained CNN model used in for the classification purpose. The final model accuracy was sufficient to carry on with our project.

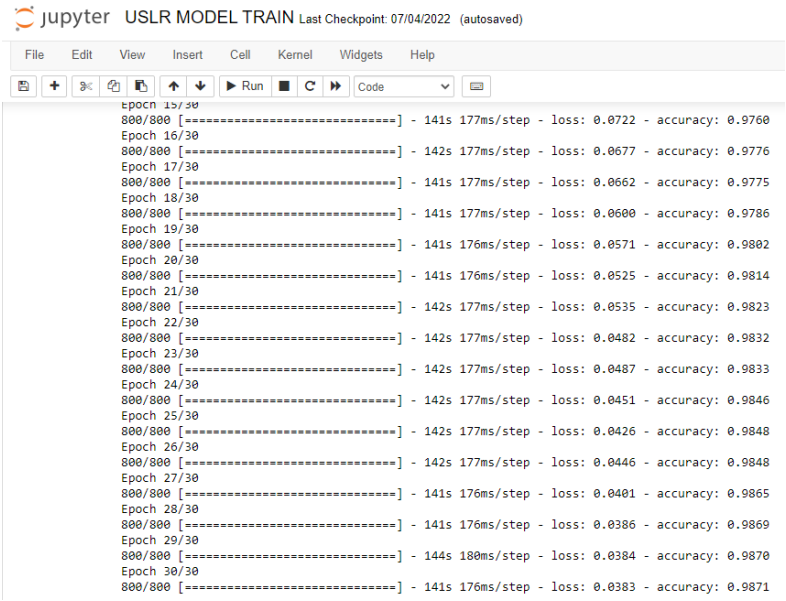

Figure 5.10: Model Accuracy

Chapter 6

System Testing and Evaluation

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. Test and evaluation is the process of resembling a system or component to requirements and specifications via testing. The results are analyzed to determine, among other things, the design's progress, performance, and maintainability. We've discussed the methodologies and test cases that we used to put our application through its paces in this chapter. System testing is critical for any system because it ensures that the software will perform as expected and will function properly.

6.1 GUI Testing

GUI testing is the process of ensuring that a product's graphical user interface (GUI) adheres to its specifications. This type of testing ensures that the application under test acts as expected when a user executes a specific action or documents a specific input and generates the appropriate program output. In our application, we tested the buttons to see how they responded.

6.2 Usability Testing

Usability testing is a type of testing that is performed from the perspective of an end-user to determine whether the system is easily usable. Usability testing is the practice of evaluating the ease of use of a design with a group of representative users. Usability testing assists us in determining how easy it is to use, how understandable it is, and how well it meets the needs of the users. During this testing, we ensured that our camera was functioning

properly and that the application interaction was simple for the user. We also made certain that the application is self-explanatory.

6.3 Compatibility Testing

Compatibility testing is a non-functional testing type performed on application software to ensure compatibility with various computing environments. Because different software is available in different versions, some may support libraries while others may not, it is possible for an application to encounter errors, or some may be outdated or more up-to-date versions. With the help of compatibility testing, we can evaluate the performance of our application on different OS versions. Our application is for Android 5 and up, and it works on about 98 percent of all Android devices. This application is designed specifically for Android devices. As a result, it cannot be used on the Web.

6.4 Performance Testing

In general, performance testing is a testing practice used to specify how a system functions in terms of responsiveness and stability under a specific workload. For this test, we tried to load and process big images, and it worked as expected and gave us the results we wanted.

6.5 Installation Testing

Installation testing verifies that the software application was installed correctly and is operating as expected. This is the testing phase that occurs prior to the first time an end user interacts with the application. During this testing, we installed our application on the user's end and ensured that the user had no problems using the application and that all functionalities were complete. We tried our application on different Android phones with different operating systems, and it worked perfectly.

6.6 Test Cases

A test case is a series of operations carried out on a system to see if it complies with software requirements and operates properly. For our project we have used different types of test cases that are used to measure the validity and robustness of our project.

Table 6.1: Test Case for Working of Screens

Test case ID	01
Test case Title	Working of Screen
Description	Check all the screen are working.
Test Steps	• - Run the application on mobile phone. • - Check all the screen
Expected Output	Screens are working
Status	Successful

Table 6.1 shows the test case of correct working of all screens including the user manual of the application.

Table 6.2: Test Case for Functionality of Camera

Test case ID	02
Test case Title	Functionality of Camera
Description	To check if camera is functional.
Test Steps	• - Open application. • - Chose the translate option. • - Show Hand gesture in front of the camera.
Expected Output	Camera is functional
Status	Successful

Table 6.2 shows the test case of working camera of the device that will be used for image capturing process.

Table 6.3: Test Case for Gesture Recognition

Test case ID	03
Test case Title	Gesture Detection
Description	Check if the gestures are recognized or not.
Test Steps	• - Chose the Translate option. • - Capture and send image via API for translation.
Expected Output	Gesture Recognized
Status	Successful

Table 6.3 shows the test case of working of sign that whether or not they are being classified correctly.

Table 6.4: Test Case for Learning Screens

Test case ID	04
Test case Title	Learning Screen
Description	Check all the signs are shown for learning.
Test Steps	• - Select the learn option in application. • - Check all the buttons.
Expected Output	Learning signs are shown.
Status	Successful

Table 6.4 shows the test case of working of learning screen of the application, while checking all the buttons individually.

Table 6.5: Test Case for Main Menu

Test case ID	05
Test case Title	Display Main Menu
Description	Display main screen.
Test Steps	• - Install the application on android mobile phone. • - Select the option that user wants to perform.
Expected Output	User is comfortable with UI.
Status	Successful

Table 6.5 shows the test case of working of application. This test case was used to install the application on multiple devices and check for the proper working of the whole application.

Chapter 7

Conclusions

USLR is an Urdu Sign Language Recognition System that was developed for the recognition of Urdu hand signs. The concepts used are machine learning based on neural networks and image processing. USLR's overall translation system is divided into two major modules:

- The Detection module
- Learning module

The detection module is used for alphabet detection. While the learning module is used for interactive USL. Due to the lack of a large pre-existing USL data set, we have introduced a new USL data set that consists of binary images of human hand signs in which 2 people are making signs of the 37 Urdu alphabets. In this data set, the ratio used for testing and training is 1:4. Numerous machine learning models were considered, but we determined that CNN was the most appropriate for this task. We were surprised to see that the optimal CNN design came from a simple architecture rather than a complex one. The final alphabet model came out very well.

Our mobile application provides a simple interface for uploading images and producing output. The output is displayed in the form of a string on the main screen of the mobile. We made it all possible with the personal supervision of Ms. Maria Mahmood. We hope that this application will provide all the benefits it could provide to the users.

7.1 Future Work

The Urdu Sign Language Recognition System was developed to recognize Urdu alphabets only. Our work can be enhanced to recognize words or sentences in the Urdu language, which will be a great contribution in the field of artificial intelligence.

Appendix A

Data Dictionary

TERMS	DESCRIPTION
Deep Learning	**Deep learning** model is a Artificial intelligence (AI) and machine learning techniques how people acquire specific types of information. Deep learning is an important element of data science, which covers statistics and predictive modelling.
USLR	**Urdu Sign Language Recognition** is an android based application that is used to overcome the communication gap between the signers and non-signers community. It provides ease in terms of cost effectiveness as well as provides the ease of use.
CNN	**Convolutional Neural Network** is a neural network type, that is particularly adept at processing input with a grid-like architecture, like an image. A binary representation of visual data is a digital image.

References

[1] Muskan D. "SIGN LANGUAGE RECOGNITION" in Summer Research Fellowship Programme of India's Science Academies, 2017.

[2] Faryal A. "Vision Based Sign Language Identification System Using Facet Analysis" in Bachelor Thesis, 2013.

[3] Shivashankara S., Dr.Srinath S. "American Sign Language Recognition System: An Optimal Approach" in International Journal of Image, Graphics and Signal Processing, August 2018.

[4] Nabeeel S., Adnan S., Saleem A., Adnan A., Yaser D., Shoaib F., Tahir M., Inayatullah K. "A Vision Based Approach for Pakistan Sign Language Alphabets Recognition" in La Pensée, March 2014.

[5] Mayuresh K., Shireen M., Aniket M. "Sign Language Recognition System" in International Journal of Scientific & Engineering Research, Volume 4, Issue 12, December-2013.

[6] Daniel S., Philippe D., Hermann N, Sara M., Andy W. "Hand in hand: Automatic Sign Language to English Translation" in ACL Anthology, April 2012.

[7] Dario Radečić. "TensorFlow for Computer Vision" in Towards Data Science, November 2021.

[8] Ali I., Abdul R., Irfan A., Aamir H., Sharaiz S., Tausif-urRehman. "Dataset of Pakistan Sign Language and Automatic Recognition of Hand Configuration of Urdu Alphabet through Machine Learning" in Elsevier Inc., June 2021.