SIGN SENSE

PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Computer Application.

Submitted by

Dan Emmanuel Antony	**U2145014**
Emmanuel Dannie Densil Fernandez	**U2145015**
Job G Panicker	**U2145016**
Sharon Jacob Mathew	**U2145029**
Sunil S	**U2145032**

Guided By

Mrs.Ancy Michael

DEPARTMENT OF COMPUTER SCIENCE

FATIMA MATA NATIONAL COLLEGE(AUTONOMOUS)

KOLLAM-691001

2021-2024

DEPARTMENT OF COMPUTER SCIENCE

FATIMA MATA NATIONAL COLLEGE(AUTONOMOUS)

KOLLAM-691001

Certificate

Certified that this project titled Sign Sense is a bonafide record of the work done by **Sunil S** (Reg. No. U2145032),**Job G Panicker** (U2145016),**Sharon Jacob Mathew**(U2145029),**Dan Emmanuel Antony** (U2145014), **Emmanuel Dannie Densil Fernandez** (U2145015) under my supervision, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Computer Application.

Miss Ancy Micheal	**Mrs. Neetha B.S**
Project Guide	**Project Coordinator**
Assistant Professor	**Assistant Professor**
Department of Computer science	Department of Computer Science

ExternalExaminer

ACKNOWLEDGEMENT

We would like to express our heartfelt thanks to our project guide Miss Ancy Micheal ,for her help and support.

We are also thankful to Dr Cynthia Catherine Michael, Principal of Fatima Mata National College for giving us an opportunity to present our project.

With a profound sense of gratitude, we would like to express our sincere gratitude to Mrs.Neetha BS, Head of the Department and our Project Coordinator for their valuable guidance and encouragement in pursuing this project.

Our sincere thanks are alsoextended toDr.TitusAR,coordinator of Self Finance,as well as to all the teachers of the Department of Computer Science,my family and to my friends for their help and support.

Above all,we thank God for the immense grace and blessings at all stages of our project work.

ABSTRACT

Sign language serves as a crucial means of communication for individuals with hearing impairments, offering a visual language that relies on hand gestures, facial expressions, and body movements. This project aims to develop a Sign Language Recognition Application using Python, leveraging machine learning techniques to interpret and translate sign language gestures into meaningful text or spoken language.

CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

The world is hardly live without communication, no matter whether it is in the form of texture, voice or visual expression. The communication among the deaf and dumb people is carried by text and visual expressions. Gestural communication is always in the scope of confidential and secure communication. Hands and facial parts are immensely influential to express the thoughts of human in confidential communication. Sign language /S learned by deaf and dumb, and usually it is not known to normal people, so it becomes a challenge for communication t between a normal and hearing impaired person. Its strike to our mind to bridge the between hearing impaired and normal people to make the communication easier.Sign language recognition (SLR) system takes an input expression from the hearing Impaired person gives output to the normal person in the form text or voice.

1.1 PROBLEM STATEMENT

The lack of efficient and accessible sign language interpretation tools poses challenges for the deaf and hard of hearing community in communicating effectively with non-signers

1.2 SCOPE AND RELEVANCE OF THE PROJECT

This project focuses on developing a mobile application capable of real-time sign language detection and interpretation. The application will use computer vision and machine learning algorithms to recognize and translate sign language gestures into text or spoken language. The relevance of this project lies in its potential to enhance communication accessibility for the deaf and hard of hearing community in various settings, including educational, professional, and social environments.

1.3 Main Objective

The main objective of this project is to create a user-friendly and accurate sign language detection application that can be easily accessed and utilized by both signers and non-signers, thereby promoting inclusive communication and fostering greater accessibility for the deaf and hard of hearing community

CHAPTER-2

SYSTEM REQIUREMENTS AND SPECIFICATIONS

This is the description on the development, and production of the proposed system. How this project is executed, the tools and cost needed, and the benefits of the newly proposed System.

2.1 EXISTING SYSTEM

Discuss the current methods or applications available for sign language detection. This could involve any software, hardware, or combination thereof that is currently in use for this purpose. Mention any limitations or shortcomings of the existing system

2.1.1 LIMITATIONS OF THE EXISTING SYSTEM

Limitations of Existing System: Discuss shortcomings, such as lack of accuracy, limited vocabulary recognition, or slow response time.

2.2 PROPOSED SYSTEM

Outline your idea for a new sign language detection application. Describe its features, functionality, and how it addresses the limitations of the existing systems. This could include advancements in technology, user interface improvements, or additional features for better accuracy and usability

2.2.1 ADVANTAGES OF PROPOSED SYSTEM

Highlight benefits like improved accuracy, expanded vocabulary recognition, faster response time, and potentially additional features like gesture recognition.

2.3 FEASIBILITY STUDY

It is a study toevaluatethefeasibilityoftheproposedprojectorsystem.Itisthemeasureofthe product in terms of how much beneficial the product development will be for the organization in a practical point of view. It includes the following processes:

- Information collection.
- Information assessment.
- Report writing.
- General Information.

There are4 types of feasibility study conducted for a project:

2.3.1 Operational Feasibility

Analyze if the proposed system can be effectively integrated into existing workflows and if users can easily adapt to it.

2.3.2 Technical feasibility

Assess if the required technology is available or can be developed to support the proposed system.

2.3.3 Economic Feasibility

Determine if the benefits of the proposed system outweigh the costs associated with its development, implementation, and maintenance

2.2.4 LEGAL FEASIBILITY

Consider any legal or regulatory issues, such as privacy concerns or compliance with accessibility standards, that may impact the development and deployment of the system.

CHAPTER-3

SYSTEM DESIGN

System design involves the collection of requirements needed for the development of the project and converting it into a way of depicting the flow of the project.
System design consists of three steps:

- Drawing of the expanded system data flow charts to identify all the processing functions required.
- The allocation of the equipment and the software to be used.
- The identification of the test requirements for the system.

Characteristics of Design

- A design should exhibit a hierarchical organization that makes intelligent use of
- control among components of the software.
- A design should be modular, that is, the software should be logical.
- A design should contain distinct and separable representation of data and procedure.
- A design should lead to interface that reduce the complexity of the connections
- between modules and with the external environment

3.1 DATA FLOW DIAGRAM

Data Flow Diagram (DFD)representing a system at any level of detail with a graphic network of symbols showing data flows, data stores, data processes, and data sources. The purpose of DFD is to provide a semantic bridge between users and system developers. The diagram is the basis of structured system analysis. A level 0 DFD, also called a fundamental system model or a context model represents the entire software elements as a single bubble with input and output indicated by incoming and outgoing arrows respectively. Additional process and information flow parts are represented in the next level i.e., Level 1 DFD. Each of the processes represented at Level 1 are sub functions of overall system depicted in the context model. Any processes, which are complex in Level 1, will be further represented into sub functions in the next level, i.e., in level 2. Data flow diagrams illustrate how data is processed by a system in terms of inputs, and outputs. Represent major components or functions with Circles. Actions for input by a user or a system go in Rectangular Boxes. Data bases are represented by Parallel in es enclosing a phrase corner.
Given below are the Data Flow Diagrams for the proposed project.

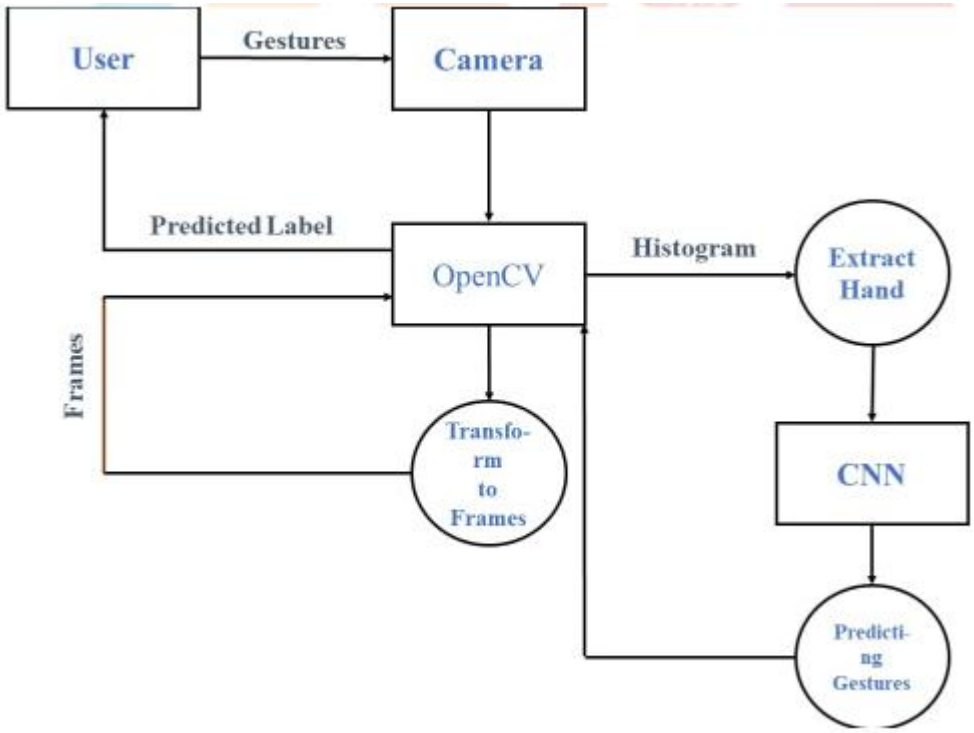


Fig 3.1 Dataflow Diagram

3.2 FLOW CHART

The sign language detection application begins when the user initiates it on their device, activating the process of capturing live video input from the device's camera. This video input undergoes preprocessing steps, including noise reduction and normalization, to ensure optimal data quality for subsequent analysis. From the preprocessed video, relevant features such as hand shape, movement trajectory, and orientation are extracted. These features serve as input for a trained machine learning model specifically designed for sign language recognition. Utilizing this model, the application processes the extracted features and accurately classifies the sign language gestures in real-time. The recognized gestures and their corresponding meanings are promptly displayed to the user, providing immediate feedback on the application's performance and enabling seamless communication. Once the process concludes, the user can choose to continue interacting with the application or exit as desired, ensuring a user-friendly and efficient experience.

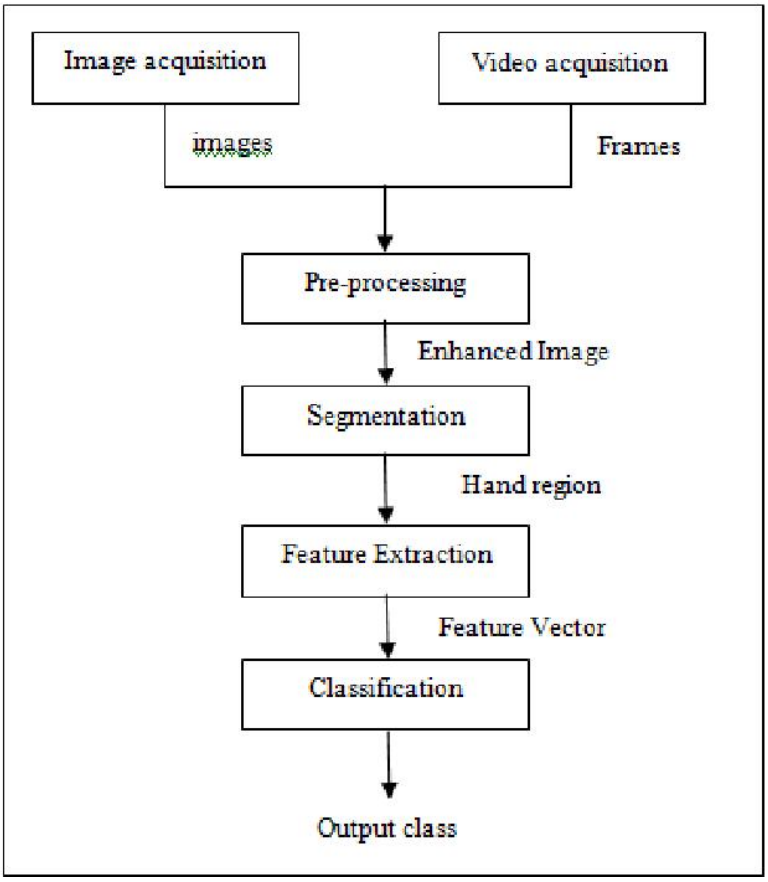


Fig 3.2 Flow Chart

3.3 USECASE DIAGRAM

In a use case diagram, we can depict various actors interacting with the system to achieve specific goals. In the context of a sign language detection application, the primary actor would be the "User" who interacts directly with the application. The user's goals may include initiating the application, performing sign language gestures, and receiving feedback on the recognized gestures. Additionally, there could be secondary actors such as "Administrator" responsible for managing the application settings and "Developer" involved in maintaining and updating the application. Each actor's interactions with the system, represented by use cases, would include actions like starting the application, capturing video input, preprocessing the data, recognizing gestures, providing feedback, and managing application settings. These interactions collectively form a use case diagram, illustrating the various scenarios in which the actors interact with the sign language detection application to achieve their respective goals.

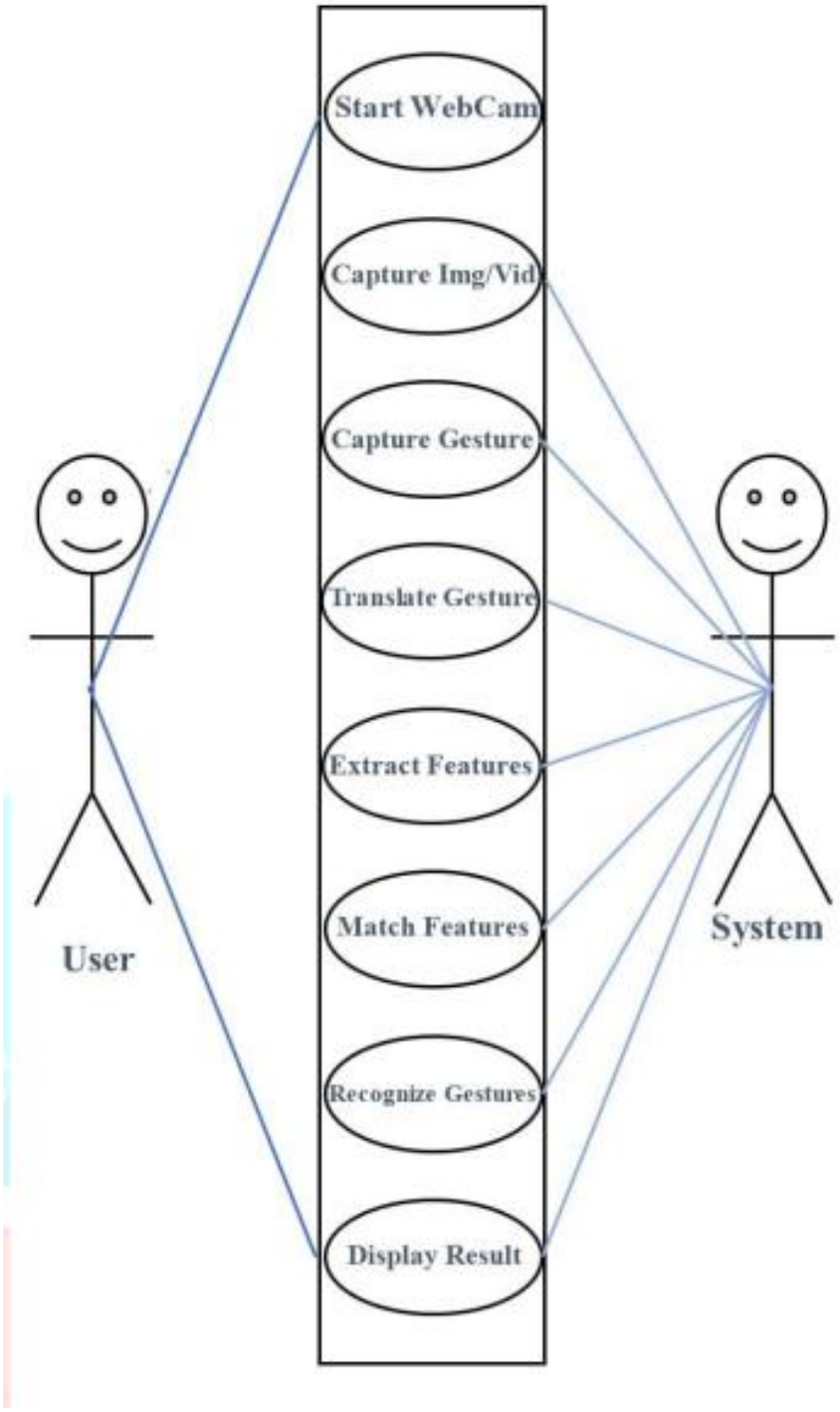


Fig 3.3 Usecase Diagram

3.4 STATECHART DIAGRAM

A state chart diagram illustrates the different states that an object or system can transition through in response to events. For a sign language detection application, the main states could include "Idle," "Capturing Video," "Processing," "Recognizing Gestures," and "Displaying Feedback." The application starts in the "Idle" state when it is initiated by the user. Upon receiving a command to start capturing video input, it transitions to the "Capturing Video" state, where it continuously collects video data from the device's camera. Once the video data is captured, the application moves to the "Processing" state, where it preprocesses the data to enhance its quality for gesture recognition. Upon completion of preprocessing, the application enters the "Recognizing Gestures" state, where it utilizes machine learning algorithms to classify the sign language gestures in real-time. If a gesture is successfully recognized, the application transitions to the "Displaying Feedback" state, where it provides immediate feedback to the user regarding the recognized gesture and its meaning. After displaying feedback, the application returns to the "Idle" state, ready to receive new commands or gestures. This state chart diagram captures the various states and transitions within the sign language detection application, illustrating its dynamic behavior in response to user interactions and system events

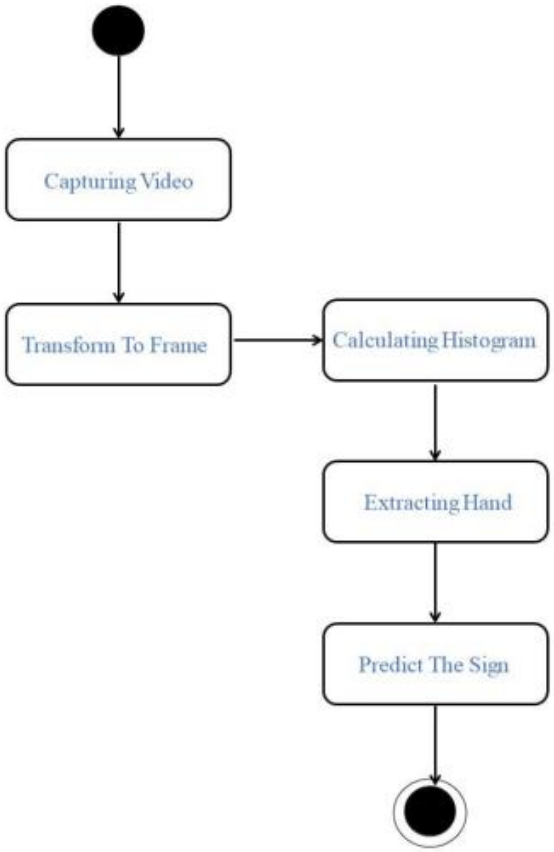


Fig 3.4 State Chart Diagram

3.5 SEQUENCE DIAGRAM

Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.In simpler words, a sequence diagram shows **how different parts of a system work in a 'sequence' to get something done**.Sequence diagrams are commonly used in software development to illustrate the behavior of a system or to help developers design and understand complex systems. They can be used to model both simple and complex interactions between objects, making them a useful tool for software architects, designers, and developers.

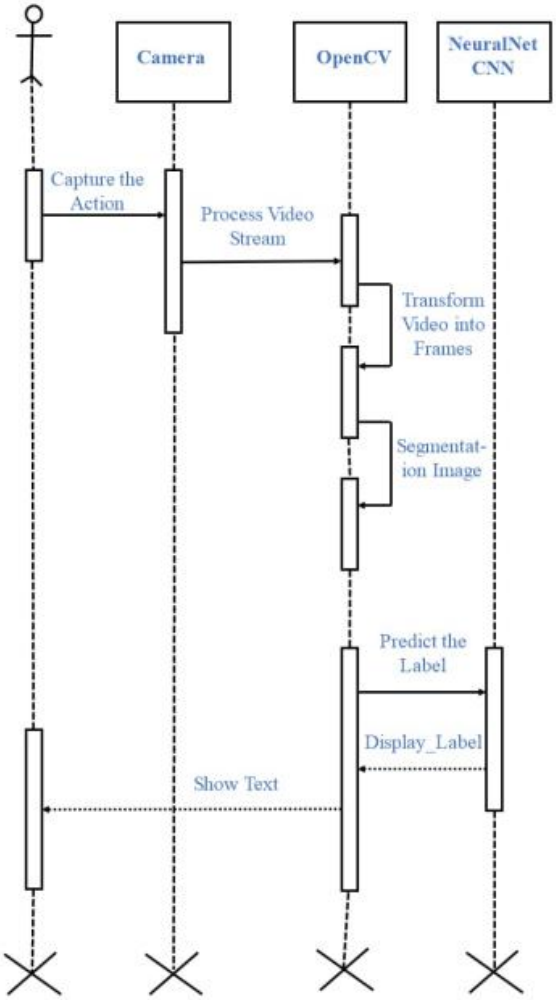


Fig 3.5 Sequence Diagram

3.6 MODULE DESCRIPTION

- User Interface Module*: This module handles the interaction between the user and the application. It includes components such as buttons, menus, and visual displays that allow users to initiate the application, adjust settings, and receive feedback on recognized gestures.

- Video Input Module*: Responsible for capturing video input from the device's camera. This module accesses the camera hardware, retrieves the video stream, and provides it to other modules for further processing.

- Preprocessing Module*: Processes the raw video input to enhance its quality and suitability for gesture recognition. This module may perform tasks such as noise reduction, normalization, and image enhancement to improve the accuracy of gesture recognition algorithms.

- Gesture Recognition Module*: Utilizes machine learning algorithms to recognize sign language gestures from the preprocessed video input. This module analyzes features extracted from the video frames and classifies the gestures in real-time, determining their meanings based on a trained model.

- Feedback Module*: Provides immediate feedback to the user based on the recognized gestures. This module displays the recognized gestures and their corresponding meanings to the user through the application's user interface, ensuring smooth communication and interaction.

- Database Interface Module*: Manages interaction with the application's database, which contains data related to sign language gestures and their meanings. This module handles tasks such as storing new gesture data, retrieving training data for the machine learning model, and updating the database with user feedback for continuous improvement.

Each module plays a crucial role in the sign language detection application, contributing to its overall functionality and usability. Together, they enable the application to accurately recognize sign language gestures in real-time and provide meaningful feedback to users, facilitating communication for individuals who are deaf or hard of hearing.

3.7 INPUT DESIGN

Input design in the sign language detection application involves designing the user interface and mechanisms for users to input sign language gestures effectively. Here's an overview of input design considerations:

- Video Input: The primary input method is through video captured from the device's camera. The input design should ensure that the camera feed is clear, well-lit, and properly framed to capture the user's sign language gestures accurately.

- User Interface: Design an intuitive and user-friendly interface that allows users to interact seamlessly with the application. This includes incorporating buttons, menus, and controls for initiating the application, adjusting settings, and providing feedback.

- Gesture Recognition: Implement mechanisms for users to perform sign language gestures comfortably within the camera's view. Provide visual cues or instructions to guide users on how to position their hands and perform gestures effectively for optimal recognition.

- Accessibility: Ensure that the input design accommodates users with different abilities and preferences. Provide options for adjusting input settings, such as camera angle, sensitivity, and gesture recognition thresholds, to accommodate varying user needs.

- Feedback Mechanism: Incorporate real-time feedback mechanisms to inform users of the application's recognition of their gestures. This could include visual indicators, auditory cues, or textual feedback displayed on the user interface.

- Customization: Offer customization options for users to personalize their input experience. This could include customizable gestures, gesture libraries, and user profiles to cater to individual preferences and communication styles.

- Error Handling: Implement robust error handling mechanisms to address input errors and inaccuracies. Provide clear error messages and guidance to help users correct their gestures and improve recognition accuracy.

- Testing and Iteration: Continuously test and iterate on the input design through user feedback and usability testing. Identify areas for improvement and refine the input mechanisms to enhance the overall user experience and effectiveness of gesture recognition.

By focusing on these input design considerations, the sign language detection application can provide users with an intuitive and accessible interface for inputting sign language gestures effectively and accurately.

3.8 OUTPUT DESIGN

- Output design in the sign language detection application involves presenting the results of gesture recognition to the user in a clear and understandable manner. Here are some key aspects of output design:

- Visual Feedback: Display the recognized sign language gestures visually to the user. This could involve showing an animated representation of the detected gesture on the screen, alongside any relevant text or symbols indicating its meaning.

- Textual Feedback: Provide textual feedback to accompany the visual representation of the recognized gesture. This helps reinforce the meaning of the gesture and ensures accessibility for users who may rely on text-based communication.

- Auditory Feedback: Incorporate auditory cues or feedback to accompany the visual and textual representations of the recognized gestures. This can enhance the user experience and provide additional reinforcement of the communication.

- Real-Time Updates: Ensure that the output is provided to the user in real-time, immediately after the gesture is recognized. This allows for seamless communication and interaction, without delays or interruptions.

- Customization Options: Offer customization options for the output design to accommodate individual user preferences and accessibility needs. This could include adjustable font sizes, color schemes, and audio settings.

- Feedback Consistency: Maintain consistency in the feedback provided to the user across different gestures and interactions. This helps users develop familiarity with the application and builds confidence in the accuracy of the gesture recognition system.

- Error Handling: Implement error handling mechanisms to address any inaccuracies or misunderstandings in the recognized gestures. Provide clear guidance on how users can correct errors or improve recognition accuracy.

- User Interaction: Enable user interaction with the output, allowing them to interact with the recognized gestures and provide feedback on their accuracy or relevance. This fosters engagement and participation in the communication process.

By focusing on these aspects of output design, the sign language detection application can provide users with clear, understandable, and accessible feedback on the recognized sign language gestures, facilitating effective communication for individuals who are deaf or hard of hearing.

CHAPTER-4

SYSTEM ENVIRONMENT

The Hardware environment refers to the physical components of a computing system or device including the machinery,peripherals and physical infrastructure required for the system to operate.The hardware environment forms the foundation on which the software applications run and interact providing the necessary resources and interfaces for the users and software programs to interact with the system.

4.1 SOFTWARE AND HARDWARE SPECIFICATIONS

Hardware specifications

- Camera: Good quality,3MP
- Ram: Minimum 8GB or higher
- GPU: 4GB dedicated
- Processor: Intel Pentium 4 or higher
- HDD/SSD: 10GB or higher
- Monitor: 15” or 17” color monitor
- Mouse: Scroll or Optical Mouse or Touch Pad
- Keyboard: Standard 110 keys keyboard

Software specifications

- Python (Version 3.7.13)
- JAVA
- IDE (PyCharm)
- NumPy (version 1.19.2)
- Cv2 (OpenCV) (version 3.4.2)
- Keras (version 2.3.1)
- TensorFlowLite (as keras uses TensorFlow in backend and for image preprocessing) (version 2.8.0)
- PYCHARM (Version 2022.1)

4.2 TOOLS AND PLATFORMS

Front End Tool

For the front-end development of the sign language detection application, a suitable tool would be React.js. React.js is a popular JavaScript library for building user interfaces, offering a component-based architecture that enables developers to create interactive and responsive UIs efficiently. With React.js, developers can design the user interface components of the application, such as buttons, menus, and visual displays, ensuring a smooth and intuitive user experience

Back End Tool

As for the back-end development, Python with Flask would be a suitable choice. Flask is a lightweight and flexible web framework for Python, ideal for building web applications with minimal overhead. With Flask, developers can implement the server-side logic of the sign language detection application, including handling requests, processing data, and interfacing with databases. Python's rich ecosystem of libraries also makes it well-suited for implementing machine learning algorithms for gesture recognition.

Platform

In terms of platform, deploying the application on the cloud using a service like Google Cloud Platform (GCP) would be advantageous. GCP offers scalable and reliable cloud infrastructure services, including virtual machines, storage, and networking, enabling developers to deploy and manage the sign language detection application with ease. Additionally, GCP provides machine learning services such as TensorFlow and AI Platform, which can be leveraged for training and deploying machine learning models for gesture recognition. Overall, using React.js for the front end, Python with Flask for the back end, and deploying on Google Cloud Platform would provide a robust foundation for developing and deploying the sign language detection application

CHAPTER 5
SYSTEM IMPLEMENTATION

Implementation phase is the phase, which involves the process of converting a new system design into an operation alone. It is the key stage in achieving a successful new system. Implementation is the stage if the project,where the theoretical design is turned into a working system.At this stage the main workload,the greatest up heal and the major impact on existing practices shift to user department. If the implementation stage is not planned and controllers carefully, it can cause chaos. The implementation stage is a system project in its own right. It involves careful planning, investigation of the current system and its constraints on the implementation, design methods to achieve the change-over procedures, and evaluation of changeover methods.

The implementation plan consists of the following steps:

- Testing the developed system within the Sample data
- Detection and correction of errors
- Making necessary changes in the system
- Training and involvement of user personnel
- Installation of software utilities.

The implementation phase is less creative than system design. A system project may be dropped at any time prior to the implementation, although it becomes more difficult when it goes to the design phase. The final report to the implementation phase includes procedural charts record layout and a workable plan for implementing the candidate system. Implementation is used to the process of converting a new or revised system design into an operation alone.Conversion is one aspect of implementation.Several procedures are unique to the implementation phase.

5.1 CODING

Computerprogrammingistheprocessofdesigningandbuildinganexecutablecomputer program for accomplishing a specific computing task. Programming involves tasks such as analysis,generating algorithms,profilingalgorithms'accuracyandresourceconsumption,and the implementation of algorithms in a chosen programming language (commonly referred to as coding).The source code of a program is written in one or more programming languages.
The purpose of programming is to find a sequence of instructions that will automate the performance of a task for solving a given problem. The process of programming thus often requires expertise in several different subjects,including knowledge of the application domain, specialized algorithms, and formal logic.

5.1.1 Code validation and optimization

The major decisions of a validation stage are concerned with handling errors and distribution of data. There are various ways of handling errors open to the designer, which includes rejection of the item of input or processing the next item, writing error record and signalling the appropriate message to the user. Error procedures must be specified in details showing decisions,action and expectations.In this project few alternatives arearranged.

5.1.2 Coding Standards

Coding standards can be understood as a series of procedure for a specific programming language that determines the programming style, procedures, methods, for various aspects of the program written in that language.A coding standard ensures that all developers writing the code in a particular language write according to the guidelines specified. This makes the code easy to understand and provides consistency in the code.This is because it positively impacts the quality of the system. The good characteristic of a good code:

- Reliable
- Maintainable
- Efficient

5.2 DEBUGGING

Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system. Debugging tactics can involve interacting debugging, control flow analysis, unit testing, integration testing, log file analysis, monitoring at the application or system, memory dumps and profiling.

UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly,and that program inputs produce valid outputs.All decision branches and internal code flow should be validated. It is the testing of individual software units.It is done after the completion of an individual unit before integration. This is

Structural testing,that relies on knowledge of its construction and is invasive.Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.Unittestsensurethateachuniquepathofabusinessprocessperformsaccurately to the documented specifications and contains clearly defined inputs and expected results.

CHAPTER-6

SYSTEM TESTING

Testing is the process of verifying a system with the purpose of identifying any errors, gaps or missing requirement versus the actual requirement. Testing is broadly categorized into two types-functional testing and non-functional testing. When to start test activities: Testing should be started as early as possible to reduce the cost and time to rework and produce software that is bug-free so that it can be delivered to the client. However, in Hardware Product Development Life Cycle, testing can be started from the Requirements Gathering phase and continued till the software is out there in productions. It also depends on the development model that is being used. For example, in the Water fall model, testing starts from the testing phase which is quite below in the tree;but in the model,testing is performed parallel to the development phase.

6.1 INTEGRATION TESTING

Integration tests are designed to test integrated components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.Integrationtestsdemonstratethatalthoughthecomponentswere individually satisfaction ,as shown by successfully unit testing, the combination of components is correct and consistent.Integrationtestingisspecificallyaimedatexposingtheproblemsthat arise from the combination of components.

6.2 SYSTEMT ESTING

System Testing is the testing of a complete and fully integrated product. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system. System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective.

6.3 BLACK BOX TESTING

It is also called as Behavioural/Specification-Based/Input-Output Testing. Black Box Testing is a software testing method in which testers evaluate the functionality under test without looking at the internal code structure.This can be applied to every level of testing such as Unit, Integration, System and Acceptance Testing. Testers create test scenarios/cases based on software requirements and specifications.So, it is AKA Specification Based Testing. Tester performs testing only on the functional part of an application to make sure the behaviour of the system is as expected. So, it is AKA Behavioural Based Testing.

Black Box Testing Techniques: 1.Equivalence Partitioning, 2.Boundary Value Analysis 3.Decision Table , 4.State Transition.

6.4 WHITE BOX TESTING

It is also called as Glass Box, Clear Box, and Structural Testing. White Box Testing is based on system's internal code structure. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. This testing usually done at the unit level.

White Box Testing Techniques:

1. Statement Coverage,2.Branch Coverage,3.Path Coverage.

CHAPTER-7

SYSTEM SECURITY MEASURES

System security encompasses all facets of accessing information assets. From software /hardware updates to modifications-security is a key component to a device operating at its optimum. These best practices help to mitigate various security concerns. We didn't ask any agree conditions .We didn't collect any sensitive data that intrude the privacy of the user.

CHAPTER-8

SCOPE AND FUTURE ENHANCEMENT

The scope of a sign language detection application encompasses real-time recognition of sign language gestures from video input, providing immediate feedback to users regarding the recognized gestures and their meanings. Key features include capturing video input, preprocessing the data to enhance accuracy, utilizing machine learning algorithms for gesture recognition, and presenting the results to users through an intuitive user interface. Additionally, the application can integrate with databases to expand its gesture recognition capabilities and support for different sign languages, as well as incorporate accessibility features to cater to users with diverse needs.

In terms of future enhancements, the application could benefit from improvements in accuracy and efficiency through ongoing optimization of machine learning algorithms and preprocessing techniques. Integration with emerging technologies such as augmented reality (AR) or virtual reality (VR) could enhance the user experience by providing immersive feedback on recognized gestures. Furthermore, expanding the application's capabilities to support additional languages and dialects of sign language would broaden its accessibility and impact, making it more useful for a wider range of users. Continuous user feedback and usability testing would be essential for identifying areas of improvement and ensuring that future enhancements align with the evolving needs of the user community.

CHAPTER-9
CONCLUSION

In conclusion, the development of a sign language detection application holds significant promise in improving communication accessibility for individuals who are deaf or hard of hearing. By leveraging technologies such as machine learning and computer vision, this application can accurately recognize sign language gestures in real-time and provide immediate feedback to users, facilitating seamless communication. The scope of the application includes capturing video input, preprocessing data, utilizing machine learning algorithms for gesture recognition, and presenting results through an intuitive user interface. Future enhancements could focus on improving accuracy, efficiency, and accessibility through optimization of algorithms, integration with emerging technologies, and support for additional languages and dialects of sign language. Overall, the development and continuous improvement of a sign language detection application represent a valuable step towards creating a more inclusive and accessible society for all.

CHAPTER-10
REFERENCE

- Sign Language Recognition Using Python and OpenCV - DataFlair (data-flair.training)
- Sign language recognition - Wikipedia
- sign-language-recognition-system • GitHub Topics • GitHub

CODES

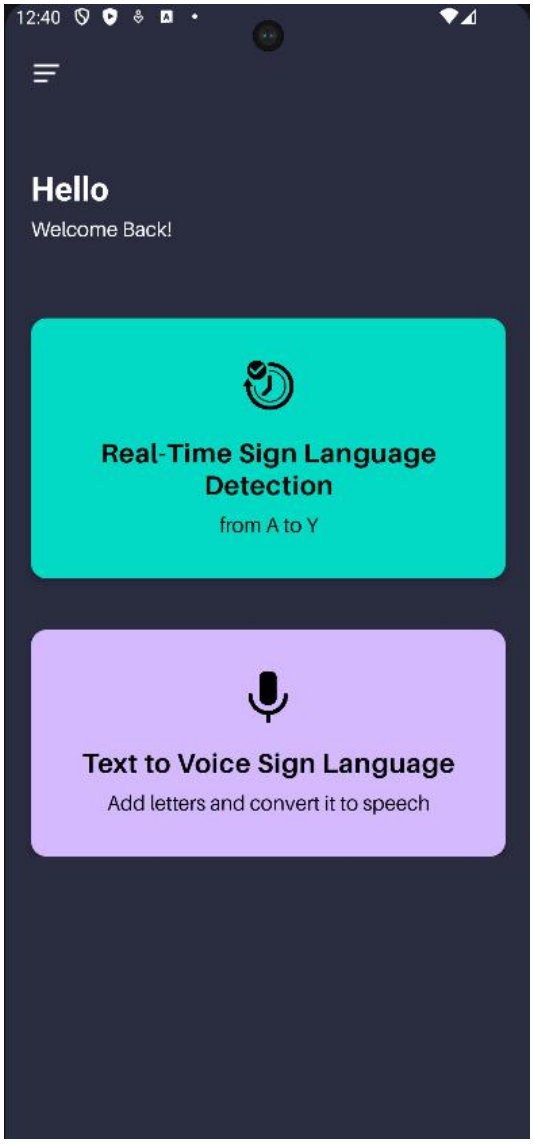
```
        if (ContextCompat.checkSelfPermission( context: CameraActivity.this, Manifest.permission.CAMERA)
                == PackageManager.PERMISSION_DENIED){
            ActivityCompat.requestPermissions( activity: CameraActivity.this, new String[] {Manifest.permission.CAMERA}, MY_PERMISSIONS_REQUEST_CAMERA
        }

        setContentView(R.layout.activity_camera);

        mOpenCvCameraView=(CameraBridgeViewBase) findViewById(R.id.frame_Surface);
        mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
        mOpenCvCameraView.setCvCameraViewListener(this);
        try{

            objectDetectorClass=new objectDetectorClass(getAssets()"Signlanguage.tflite", inputSize: "labels.txt", classification_model: 300);
            Log.d( tag: "MainActivity", msg: "Model is successfully loaded");
        }
        catch (IOException e){
            Log.d( tag: "MainActivity", msg: "Getting some error");
            e.printStackTrace();
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        if (OpenCVLoader.initDebug()){
            //if load success
            Log.d(TAG, msg: "Opencv initialization is done");
            mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }
        else{
            //if not loaded
            Log.d(TAG, msg: "Opencv is not loaded. try again");
            OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION, AppContext: this,mLoaderCallback);
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        if (mOpenCvCameraView !=null){
            mOpenCvCameraView.disableView();
```

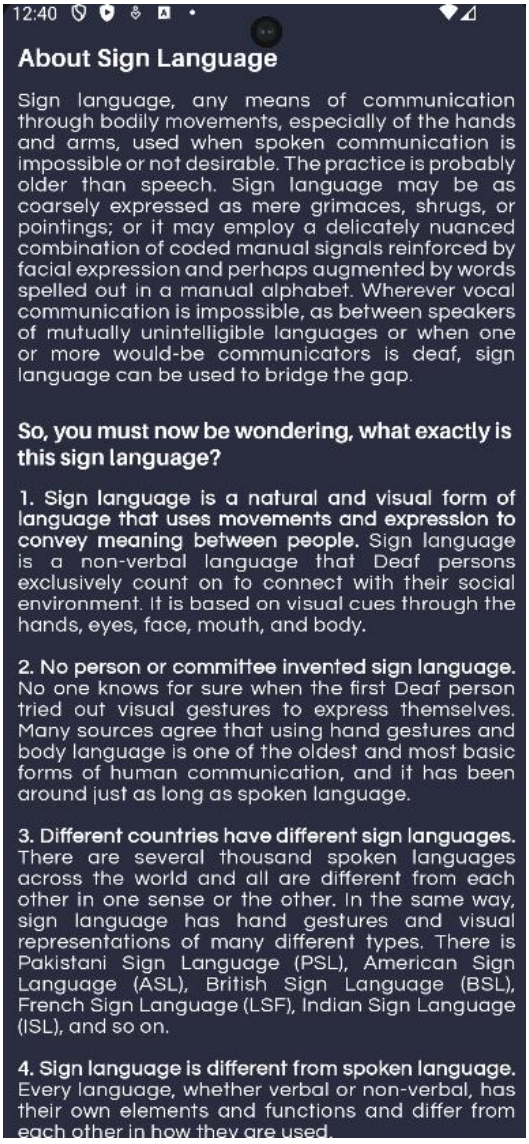
```
import org.opencv.android.OpenCVLoader;

import java.io.IOException;

3 usages
public class MainActivity extends AppCompatActivity {
    static {
        if(OpenCVLoader.initDebug()){
            Log.d( tag: "MainActivity: ", msg: "Opencv is loaded");
        }
        else {
            Log.d( tag: "MainActivity: ", msg: "Opencv failed to load");
        }
    }

    2 usages
    private Button camera_button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        camera_button=findViewById(R.id.camera_button);
        camera_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext: MainActivity.this,CameraActivity.class).addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIV
            }
        });

    }
}
```

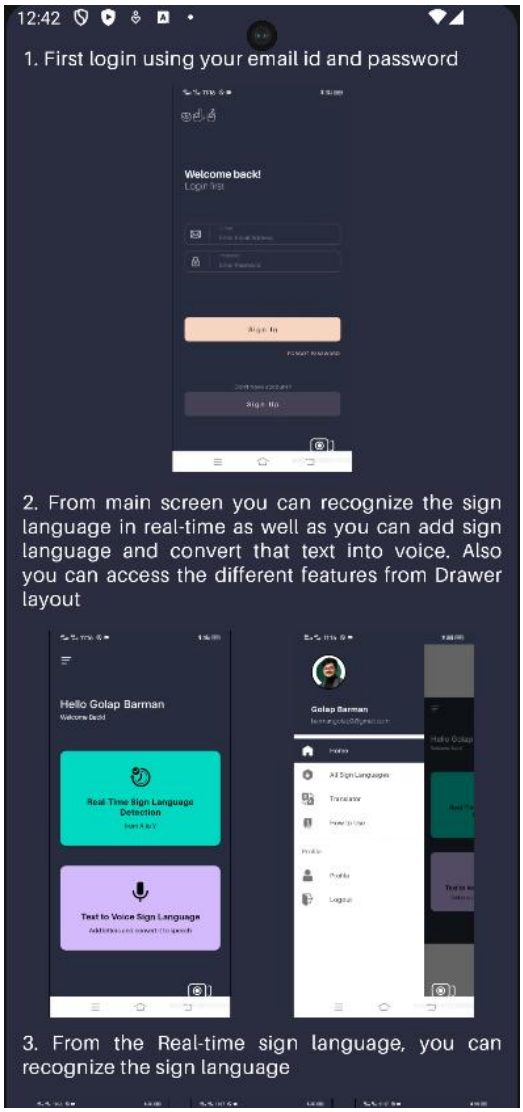
```
float[][][]boxes =new float[1][10][4];

float[][] scores=new float[1][10];

float[][] classes=new float[1][10];



output_map.put( k: 0,boxes);
output_map.put( k: 1,classes);
output_map.put( k: 2,scores);


interpreter.runForMultipleInputsOutputs(input,output_map);


Object value=output_map.get(0);
Object Object_class=output_map.get(1);
Object score=output_map.get(2);


for (int i=0;i<10;i++){
    float class_value=(float) Array.get(Array.get(Object_class, index: 0),i);
    float score_value=(float) Array.get(Array.get(score, index: 0),i);

    if(score_value>0.5){
        Object box1=Array.get(Array.get(value, index: 0),i);
        // we are multiplying it with Original height and width of frame

        float y1=(float) Array.get(box1, index: 0)*height;
        float x1=(float) Array.get(box1, index: 1)*width;
        float y2=(float) Array.get(box1, index: 2)*height;
        float x2=(float) Array.get(box1, index: 3)*width;

        if (y1 < 0) {
            y1=0;
        }
        if (x1<0){
            x1=0;
        }
        if(x2>width){
```

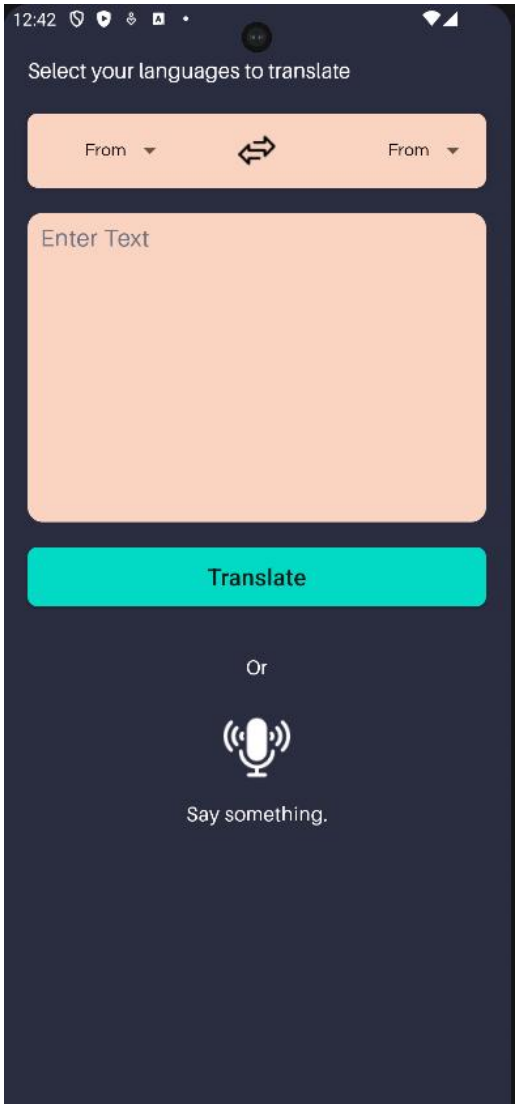
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="Sign Sense"
        android:supportsRtl="true"
        android:theme="@style/Theme.SignSense"
        tools:targetApi="31">

        <activity
            android:name=".objectDetectorClass"
            android:exported="false"
            android:label="objectDetectorClass"
            android:theme="@style/Theme.SignSense.NoActionBar" />

        <activity
            android:name=".CameraActivity"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-feature android:name="android.hardware.camera" android:required="false"/>
    <uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>
    <uses-feature android:name="android.hardware.camera.front" android:required="false"/>

</manifest>
```

SCREENSHOTS






12:42
1. First login using your email id and password
Welcome back!
2. From main screen you can recognize the sign language in real-time as well as you can add sign language and convert that text into voice. Also you can access the different features from Drawer layout
Hello Golap Barman
Real Time Sign Language Detection
Text to Voice Sign Language
Golap Barman
3. From the Real-time sign language, you can recognize the sign language


12:42
Select your languages to translate
From
From
Enter Text
Translate
Or
Say something.