SIGN LANGUAGE TRANSLATION USING PYTHON

A Project Report

Submitted in partial fulfilment of requirements for the award of the degree of

BACHELOR OF TECHONOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by:

N. Hima Sai (218T1A1237)

K. Navyatha (218T1A1227)

Sk. Galib Shaheed (218T1A1245)

K.V. Sai Krishna (228T5A1205)

Under the esteemed guidance of

Mrs. P. Kamakshi M.Tech, (Ph.D)

Assistant Professor, IT

DEPARTMENT OF INFORMATION TECHNOLOGY

DHANEKULA INSTITUTE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

(Approved by AICTE-New Delhi, Affiliated to JNTU-Kakinada)

NAAC Accredited & an ISO 9001-2015 Certified Institution

GANGURU, VIJAYAWADA, INDIA – 521139

APRIL 2025

DHANEKULA INSTITUTE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

(Approved by AICTE-New Delhi, Affiliated TO JNTU- Kakinada)

NAAC Accredited & an ISO 9001-2015 Certified Institution

GANGURU, VIJAYAWADA (INDIA)– 521139

Department of Information Technology

CERTIFICATE

This is to certify that the project titled **"Sign Language Translation using Python"** is the Bonafide work carried out by **N. Hima Sai (218T1A1237), K. Navyatha (218T1A1227), Sk. Galib Shaheed (218T1A1245), K.V. Sai Krishna (228T5A1205)** students of B.Tech in Information Technology of Dhanekula Institute of Engineering & Technology, affiliated to JNTU, Kakinada, A.P, India during the academic year 2024-25, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Information Technology and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Signature of the Guide

Mrs. P. Kamakshi, M.Tech, (Ph.D)

Assistant Professor

Signature of the HOD

Dr. K. Sandeep M.S, Ph.D

Associate Professor &

Head of the Department

Signature of External Examiner

DECLARATION

We are the students of DHANEKULA INSTITUTE OF ENGINEERING AND TECHNOLOGY, here by declare that the project entitled **"SIGN LANGUAGE TRANSLATION USING PYTHON"** submitted for the Bachelors of Technology in Information Technology degree is our original work and the project has not formed the basis for the award of any degree, diploma, fellowship or any other similar titles.

N. Hima Sai (218T1A1237)
K. Navyatha (218T1A1227)
Sk. Galib Shaheed (218T1A1245)
K.V. Sai Krishna (228T5A1205)

Place:
Date:

ACKNOWLEDGEMENT

We would like to take this opportunity to express my deep gratitude to the members who assisted us directly and indirectly for the completion of this project work.

We would like to thank **Mrs. P. Kamakshi**, Assistant Professor, the project guide for her esteemed guidance and support, especially the valuable ideas and thoughts provided during this project work. This makes us very useful in solving the problems encountered during the project work.

We would also like to thank **Mr. P. Raghuveer**, Assistant Professor, and Project Coordinator for giving opportunity to make this project a successful one.

We would like to express my immense pleasure in expressing immeasurable sense of gratitude to, **Dr. K. Sandeep**, Associate Professor & Head of the Department in Information Technology for his valuable suggestions in the completion of the project.

We whole heartedly acknowledge **Dr. Ravi Kadiyala**, Principal and also extend my thanks to all faculty members of Information Technology Department for their valuable contributions in this project.

We would like to extend my warm appreciation to all my friends for sharing their knowledge and valuable contributions in this project.

Finally, we express my deep sense of gratitude to my parents for their continuous support throughout our academic career and their encouragement in completion of this project successfully.

Signature of students

N. Hima Sai(218T1A1237)

K. Navyatha (218T1A1227)

Sk. Galib Shaheed (218T1A1245)

K.V. Sai Krishna (228T5A1205)

ABSTRACT

The Sign Language Translation System is an innovative solution designed to foster inclusive communication between hearing- and speaking-impaired individuals and the general population. At its core, the project leverages Python programming alongside MediaPipe - a robust framework for real-time hand and body pose tracking—to interpret sign language gestures with high precision. Machine learning models are trained to recognize and translate these gestures into corresponding text, ensuring accurate and fast recognition. The translated text is transmitted in real time to a mobile application via ThingSpeak, an Internet of Things (IoT) platform that facilitates seamless communication between devices.

Not only can it translate sign language into readable text, but it can also take input text from the mobile app and display corresponding sign language gestures on a laptop screen. This reverse translation allows people who are not familiar with sign language to communicate effectively with those who are hearing- or speech-impaired. The visual gestures are animated or shown as pre-recorded videos or graphical illustrations, making the interaction intuitive and clear.

The integration of IoT, machine learning, and real-time processing transforms this project into a comprehensive communication tool. It supports dynamic interaction and reduces the dependency on human translators, empowering users with greater independence. By enabling bidirectional communication, this system not only improves accessibility but also promotes inclusivity in everyday social and professional environments. Ultimately, it demonstrates how technology can bridge significant gaps in human interaction and bring communities closer together.

DHANEKULA INSTITUTE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

Department of Information Technology

VISION – MISSION – PEOs

Vision/Mission/PEOs

Institute Vision	Pioneering Professional Education through Quality
Institute Mission	Providing Quality Education through state-of-art infrastructure, laboratories and committed staff. Moulding Students as proficient, competent, and socially responsible engineering personnel with ingenious intellect. Involving faculty members and students in research and development works for betterment of society
Department Vision	To become a leading centre in Information Technology education and research, fostering innovation, technical expertise, and responsibility
Department Mission	Provide learner centric education with state-of-the-art facilities. Impart problem-solving skills to become pioneers in the global competition through trainings and various activities. Equip learners with employability and entrepreneurial skills. Promote Research environment and inculcate corporate social responsibility.
Program Educational Objectives (PEOs)	Graduates of Information Technology will: PEO1: Solve multidisciplinary problems and innovate through core IT knowledge, excelling in professional careers or higher studies. PEO2: Integrate IT across domains, demonstrate ethical professionalism, and embody environmental consciousness as competent, well-rounded individuals. PEO3: Engage in continuous learning, adapting to evolving technologies while promoting societal betterment through responsible innovation and research

DHANEKULA INSTITUTE OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Department of Information Technology

POs/PSOs

PROGRAM OUTCOMES

1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems
2	Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching sustained conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3	Design/Development of Solutions: Design solutions for complex engineering problems and design system components or process that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4	Conduct Investigations of Complex Problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5	Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6	The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7	Environment And Sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9	Individual And Teamwork: Function effectively as an individual, and as a member or a leader in diverse teams, and in multidisciplinary settings.
10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11	Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments
12	Life- Long Learning: Recognize the need for and have the preparation and ability to engage in independent and life- long learning in broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

PSO1: Design and develop the Information Technology based AI systems and software applications with technical and professional skills.

PSO2: Excel in higher studies, secure employment in diverse technology sectors, contribute to research, and entrepreneurship.

PROJRCT MAPPINGS

Batch No:	16
Project Title:	Sign Language Translation Using Python
Project Domain	Machine Learning
Type of Project	Application
Guide Name	Mrs. P. Kamakshi, M.Tech, (P.hD)
Student Roll No	**Student Name**
218T1A1237	N. Hima Sai
218T1A1227	K. Navyatha
218T1A1245	Sk. Galib Shaheed
228T5A1205	K.V. Sai Krishna

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
Improve the technical knowledge, skills, and attitudes to learn and live as a professional engineer	3	3	3	3	2	3	2	3	3	3	2	3	3	3
To analyze and make use of the tools and technologies available in the market for solving complex societal problems with ethics.	3	3	3	3	3	3	2	3	3	3	3	3	3	3
Ability to design and develop a cost effective solution for complex problems without disturbing the societal environment in a systematic approach as a team.	3	3	3	3	2	3	3	3	3	3	3	3	3	3
Total	9	9	9	9	7	9	7	9	9	9	8	9	9	9
Average of the mapping	3	3	3	3	2.3	3	2.3	3	3	3	2.6	3	3	3

Mapping Level	Mapping Description
1	Low Level Mapping with PO & PSO
2	Moderate Level Mapping with PO & PSO
3	High Level Mapping with PO & PSO

N. Hima Sai(218T1A1237)

K. Navyatha (218T1A1227)

Sk. Galib Shaheed (218T1A1245)

K.V. Sai Krishna (228T5A1205)

Mrs. P. Kamakshi

PROJECT GUIDE

MAPPING JUSTIFICATIONS

PO1: CO1, CO2, CO3 Strongly mapped for PO1, this project requires a deep understanding of mathematics, science, and engineering principles, such as signal processing, machine learning, and speech recognition algorithms. The use of modern engineering tools like Python and APIs further supports a strong application of these fundamentals.

PO2: CO1, CO2, CO3 Strongly mapped for PO2, the project thoroughly investigates the complexities of speech recognition and the challenges of converting speech into accurate text. The problem analysis includes noisy data, incomplete words, and the development of more efficient algorithms, demonstrating a strong problem-analysis approach.

PO3: CO1, CO2, CO3 Strongly mapped for PO3, this is designed to address real-world problems, such as enabling communication for hearing-impaired individuals. The design carefully considers public health and safety, and it incorporates societal, cultural, and environmental factors by making the tool universally accessible.

PO4: CO1, CO2, CO3 Strongly mapped for PO4, this project investigates complex issues related to speech recognition, using research-based methods to analyze and improve the accuracy of the system. Experiments with noisy sentences and incomplete words demonstrate a comprehensive approach to problem-solving.

PO5: CO2 Strongly mapped for PO5, this project employs Python, speech-to-text APIs, and machine learning models (such as TensorFlow or other deep learning models). It makes effective use of modern tools and resources to implement and optimize speech-to-text and sign language translation.

PO5: CO1, CO3 Moderately mapped for PO5, does not fully cover all aspects of Modern Tool Usage (e.g., tool creation and predictive modeling), it is moderately mapped due to its emphasis on the application of tools to solve real-world, complex problems in an ethical context.

PO6: CO1, CO2, CO3 Strongly mapped for PO6, this project is aimed at societal good, particularly for the hearing- impaired. It addresses the social, health, and safety aspects of communication, ensuring inclusivity and accessibility. These considerations make the project highly aligned with societal benefits.

PO7: CO1, CO2 Moderately mapped for PO7, While the project itself doesn't directly impact the environment, it does have an indirect opportunity to be energy-efficient in terms of computational resource management. However, its environmental impact is relatively minimal, and sustainability doesn't seem to be a primary focus.

PO7: CO3 Strongly mapped for PO7, embedding the values of resource optimization, social responsibility, and environmental protection into the engineering design process. It directly addresses the need for

sustainable development by promoting awareness of the societal and environmental impacts of engineering decisions.

PO8: CO1, CO2, CO3 Strongly mapped for PO8, this project should adhere to ethical principles, ensuring data privacy when handling user speech and ensuring that the translation is accurate and unbiased. Ethical responsibility is a key concern, especially since the target audience consists of vulnerable individuals (hearing-impaired users).

PO9: CO1, CO2, CO3 Strongly mapped for PO9, While the project can be worked on by individuals, but achieving the best results would require a team of multidisciplinary professionals (e.g., data scientists, speech engineers, and UX designers). Thus, it moderately emphasizes teamwork, especially in real-world deployment.

PO10: CO1, CO2, CO3 Strongly mapped for PO10, Effective communication is critical for this project, as it involves explaining complex technical processes (like speech-to-text algorithms) in a manner that can be understood by the general public, particularly the hearing-impaired. Additionally, proper documentation and effective communication of findings are vital.

PO11: CO1 Moderately mapped for PO11, project likely involves planning, coordinating, and managing resources and time. However, in the initial stages, the focus might be more on the technical development rather than extensive project management. Financial aspects like resource management and deployment could be considered as the project grows, but it's not a core emphasis from the start.

PO11: CO2, CO3 Strongly mapped for PO11, reflects strong alignment with Project Management skills such as planning, execution, and team leadership, combined with Finance skills like cost-effectiveness and resource optimization. Additionally, it emphasizes the role of engineering ethics and sustainability in creating responsible and impactful solutions. Such an approach ensures technical success while addressing broader societal needs.

PO12: CO1, CO2, CO3 Strongly mapped for PO12, it gives the rapid advancements in fields such as machine learning, speech recognition, and AI, the project encourages lifelong learning. There will be a constant need to update the system based on the latest research and advancements, fostering a culture of continuous learning.

PSO1: CO1, CO2, CO3 Strongly mapped for PSO1, it involves developing a software application that uses AI for a real- world application.

PSO2: CO1, CO2, CO3 Strongly mapped for PSO2, As the project can open up opportunities for higher education, employment in technology sectors, and contribute to the advancement of research and potential entrepreneurship in the assistive technology space.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

INTRODUCTION

1. INTRODUCTION

1.1 General Introduction

Communication plays a vital role in human interaction, enabling individuals to express their thoughts, emotions, and ideas. However, individuals with hearing or speech impairments often face significant challenges in conveying their messages or understanding others. Traditional communication aids such as interpreters, text-based systems, or sign language dictionaries have inherent limitations in terms of accessibility, real-time responsiveness, and scalability [1-4].

The Sign Language Conversion System aims to bridge this communication gap by leveraging modern technologies such as Python, MediaPipe, machine learning.[4] This project introduces a real-time, automated, bi-directional system that facilitates seamless interaction between hearing/speaking-impaired individuals and the general population. By converting sign language gestures into text and vice versa, the system promotes inclusivity, accessibility, and ease of communication [5-9].

1.2 Problem Statement

Despite rapid advancements in technology, the availability of effective communication solutions for hearing and speaking-impaired individuals remains insufficient. Many existing methods suffer from several drawbacks such as:

- Lack of Bi-Directionality: Most available solutions either convert text to sign language or sign language to text but fail to offer two-way communication.
- Dependence on Expensive Hardware: Some solutions require specialized hardware such as wearables, gesture recognition gloves, or additional sensors, making them costly.
- Need for Human Intervention: Many systems rely on trained interpreters, which are expensive, not always available, and not scalable for mass usage.
- Non-Real-Time Solutions: Traditional methods such as mobile dictionaries or static gesture recognition systems require manual input, making communication slow and inefficient.

1.3 Main Objective

To address the identified problem, the Sign Language Translation system is designed with the following objectives:

Enable Two-Way Communication

- The system ensures a seamless bi-directional communication mechanism between hearing/speaking-impaired individuals and others. Unlike existing systems that focus on either text-to-sign conversion or sign-to-text translation, this solution integrates both, enabling interactive and real-time conversations.

Convert Text Messages from a Mobile App into Animated Sign Language Gestures

- A mobile application is developed using MIT App Inventor that allows users to input text messages.
- These messages are transmitted to a laptop via ThingSpeak, a cloud-based communication platform.
- A Python script processes the text and generates corresponding sign language gestures as animations on the laptop screen.
- This feature benefits individuals who cannot hear but can understand sign language.

Detect and Interpret Sign Language Gestures into Text Using MediaPipe and Machine Learning [4]

- The system uses a laptop/PC camera to capture real-time sign language gestures.
- MediaPipe, an advanced machine-learning framework for hand and body tracking, detects key points from the video feed.
- A pre-trained machine learning model interprets the detected gestures and maps them to corresponding text messages.
- This translated text is then sent to the mobile app for display, allowing hearing-impaired individuals to communicate efficiently.

Transmit Interpreted Text to a User-Friendly Mobile App via ThingSpeak

- The detected text from the gesture recognition system is processed and sent to the mobile application using ThingSpeak.
- Users on the mobile app can read the translated text messages, enabling effective communication with sign language users.
- This cloud-based approach ensures smooth, real-time, and scalable message transmission.

Develop a Scalable, Accessible, and Cost-Effective Solution

- The entire system is designed to be cost-efficient and does not require expensive hardware such as wearables or dedicated sensors.
- The mobile application is built using MIT App Inventor, making it easily deployable on any smartphone without complex installations.

- The use of Python and MediaPipe allows the system to work on standard laptops with built-in webcams, eliminating the need for external devices.
- Since the solution relies on IoT-based ThingSpeak for communication, it ensures global accessibility and scalability for a larger audience.

The Sign Language Translation System is a pioneering approach toward inclusive and accessible communication for hearing and speaking-impaired individuals. By integrating machine learning, real-time gesture recognition, cloud-based IoT communication, and animated sign language translation, this system effectively addresses the shortcomings of traditional methods [10-19]

With its bi-directional capability, cost-effectiveness, and real-time processing, this project stands as a significant step toward breaking communication barriers, promoting inclusivity, and empowering individuals with disabilities to engage seamlessly with the world around them [20-21].

SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 Existing System and Drawbacks

1. Human Interpreters

Description: Human interpreters are trained professionals who facilitate communication between hearing-impaired individuals and the general population. They interpret spoken language into sign language and vice versa.

Drawbacks:

- Expensive: Hiring a qualified interpreter can be costly, making it unaffordable for many individuals and organizations.
- Dependent on Availability: Interpreters are not always readily available, especially in remote areas or during emergencies.
- Not Scalable: The availability of interpreters is limited, making it challenging to scale for widespread usage.
- Privacy Concerns: Individuals may not feel comfortable sharing personal information through an interpreter.

2. Static Gesture Recognition Devices

Description: Static gesture recognition devices use predefined images or patterns to identify sign language gestures. These devices capture hand positions and translate them into text [22-26].

Drawbacks:

- Limited to Predefined Gestures: These devices rely on a fixed dataset of gestures, making them inflexible to new signs or variations in signing styles.
- Inability to Recognize Dynamic Movements: Most static systems fail to capture complex gestures involving movement.
- Lack of Context Understanding: These systems do not understand contextual variations, leading to misinterpretations.
- High Maintenance: Some devices require frequent updates to remain relevant.

3. Gesture-to-Text Wearables

Description: Wearable devices such as smart gloves or armbands detect hand movements and translate them into text using embedded sensors and machine learning algorithms.

Drawbacks:

- High Cost: The production and maintenance of these wearable devices are expensive, making them inaccessible to many users.
- Requires Physical Devices: Users must wear specialized equipment, which may not always be comfortable or convenient.
- Limited Battery Life: Many wearable devices have limited battery capacity, reducing their practicality for long-term use.
- Calibration Issues: Frequent calibration is needed to ensure accuracy, adding to the complexity.

4. Sign Language Dictionaries (Mobile Apps)

Description: Sign language dictionary applications provide a database of words and their corresponding sign language representations. Users must manually search for words to learn the signs.

Drawbacks:

- Manual Input Required: Users need to type or search for words, making real-time communication difficult.
- Not Dynamic or Real-Time: These applications do not facilitate fluid conversations as they require manual selection of words.
- Limited Sign Language Support: Many apps support only a few sign languages, limiting accessibility for diverse users.
- Not Interactive: Users do not receive real-time feedback, reducing engagement and learning efficiency.

While existing methods have contributed to improving communication for the hearing and speech impaired, they come with limitations such as high costs, lack of scalability, and restricted real-time interaction. A more dynamic, cost-effective, and bi-directional solution is necessary to bridge the communication gap effectively.

2.2 Proposed System

Description

The proposed system is a real-time, bi-directional communication tool designed to enhance accessibility for individuals with hearing and speaking impairments. This system enables seamless interaction between impaired individuals and others through the integration of machine learning, computer vision, and IoT technologies. The solution comprises two primary functions: Message-to-Sign Conversion and Sign-to-Text Conversion.

In the Message-to-Sign Conversion process, users send a message via a mobile application developed using MIT App Inventor. The message is transmitted to a Python script running on a laptop through the IoT platform ThingSpeak. The script processes the received text and converts it into corresponding sign language gestures. These gestures are then displayed as animations on the laptop, allowing the hearing-impaired individual to understand the message visually [27].

In the Sign-to-Text Conversion process, a camera captures real-time sign language gestures performed by a hearing-impaired individual. The system leverages the MediaPipe framework along with a pre-trained machine learning model to detect and interpret these gestures. The recognized gestures are converted into text, which is subsequently sent to the mobile application via ThingSpeak. This functionality allows a two-way, real-time communication channel between impaired individuals and others, making interactions more natural and efficient [28-29].

2.2.1 Advantages of the Proposed System

1. Bi-Directional Communication: Enables both text-to-sign and sign-to-text conversion for effective interaction.
2. Real-Time Processing: Provides instantaneous responses, ensuring smooth communication.
3. Accessibility and Scalability: A cost-effective solution that can be easily deployed and expanded.
4. IoT Integration: Utilizes ThingSpeak for seamless data transfer between mobile and laptop systems.
5. Machine Learning-Based Recognition: Enhances accuracy in sign language interpretation.
6. Eliminates the Need for Human Interpreters: Reduces dependency on expensive and limited human resources.
7. User-Friendly Design: Simple mobile app interface and easy-to-use functionalities for wider adoption.
8. Supports Continuous Improvement: Machine learning models can be updated to recognize new signs and gestures over time.
9. Versatile Applications: Can be implemented in education, healthcare, customer service, and daily communication

2.3 Functional and Non-Functional Requirements

The development of a real-time sign language translation system requires a clear understanding of both functional and non-functional requirements. These aspects ensure that the system meets user expectations, operates efficiently, and remains scalable for future enhancements.

2.3.1 Functional Requirements

The primary functionality of the system is to enable seamless, two-way communication between individuals with hearing or speech impairments and those who do not understand sign language. This is achieved through a combination of text-to-sign and sign-to-text conversion processes. The mobile application plays a crucial role by allowing users to enter text messages, which are then transmitted to the laptop through an IoT-based platform, ThingSpeak. On receiving the message, a Python script processes the text and converts it into sign language animations, which are displayed on the screen to assist the hearing-impaired individual.

On the other hand, the system also supports the reverse communication flow, where a hearing-impaired individual can use sign language to convey a message. A camera captures real-time hand gestures, and MediaPipe—combined with a machine learning model—detects key gesture points and interprets them into corresponding text.[4] The processed text is then transmitted to the mobile app through ThingSpeak, allowing the other person to read the message. This two-way system eliminates the need for external interpreters, making communication faster, independent, and more accessible[30-32].

The mobile application, developed using MIT App Inventor, provides an easy-to-use interface, ensuring accessibility for users who may not be familiar with complex software. The app must efficiently send and receive data, ensuring that users experience a smooth and uninterrupted conversation. The accuracy of gesture recognition is another important functional aspect, as it directly affects the reliability of communication. The machine learning model should be able to differentiate between various gestures and map them correctly to textual outputs, minimizing misinterpretations.

Additionally, the system should be able to work in real-time, where delays between input and output are minimal. The integration of ThingSpeak for data transmission ensures efficient communication between devices. Users should be able to rely on the system to process their messages and gestures within seconds, making the conversation as natural as possible.

2.3.2 Non-Functional Requirements

Beyond core functionality, several non-functional aspects define the overall efficiency, usability, and effectiveness of the system.

- Performance is one of the most critical factors, as real-time processing is essential for a smooth communication experience. The text-to-sign and sign-to-text conversion processes must be optimized for speed, preventing lag in communication. The system must also handle multiple users and allow for future scalability, making it adaptable to new sign languages or additional features without significant modifications.
- Usability plays a vital role, ensuring that users of all skill levels can operate the system effortlessly. The mobile app and laptop interface should be designed with simplicity in mind, making navigation intuitive and reducing the learning curve for first-time users. The accessibility factor is particularly important for people with hearing and speech impairments, so the design must focus on clarity and ease of interaction.
- Security is another important concern, as the communication between the mobile application and the laptop should be protected from unauthorized access. Since the system involves personal conversations, securing the transmission of data through encrypted communication protocols is necessary to maintain privacy [5].
- Reliability of the system is also crucial. The gesture recognition process must work accurately across different conditions, including variations in lighting or background settings. A well-trained machine learning model should be able to function consistently without frequent misinterpretations. In addition, the system should be designed to handle potential errors gracefully, ensuring that even in case of a failure, users receive appropriate feedback or guidance on how to resolve the issue.
- Maintainability of the software is another important aspect. The architecture should be modular, allowing updates and improvements to be implemented without major redesigns. New signs or gestures should be easy to integrate into the system, ensuring that it remains relevant as sign languages evolve.

Finally, the system must be cost-effective so that individuals and institutions can adopt it without significant financial barriers. By relying on open-source technologies such as MediaPipe, TensorFlow, OpenCV, and MIT App Inventor, the development and deployment costs remain minimal, making it accessible to a wider audience. Furthermore, since the system does not require expensive hardware, it provides an affordable yet highly efficient communication tool.

2.4 Literature Survey

1. Towards Continuous Sign Language Recognition with Deep Learning

Author: Boris Mocialov, Graham Turner, Katrin Lohan, Helen Hastie

Methodology:

This paper explores deep learning techniques for continuous sign language recognition. The study identifies key challenges in recognizing sign language in continuous sequences and proposes innovative machine learning methods to improve recognition accuracy and real-time processing. The authors introduce a hybrid deep learning framework combining CNNs and RNNs to analyze temporal dependencies in sign gestures. The system has been tested on publicly available datasets, showing significant improvements over traditional methods [1].

2. Evolution of Mechanical Fingerspelling Hands for People Who Are Deaf-Blind

Author: DL Jaffe

Methodology:

The research focuses on mechanical hands designed to simulate fingerspelling, aiding communication for deaf-blind individuals. The paper traces the technological advancements in robotic fingerspelling devices and their role in improving accessibility and inclusivity. The study discusses various hardware designs, including servo-controlled robotic fingers, and their effectiveness in delivering tactile sign language communication. Real-world usability tests indicate increased efficiency in communication for users with dual impairments [2].

3. A Deep Learning-Based Approach for Real-Time Sign Language Gesture Recognition

Authors: Rajesh Kumar, Pooja Sharma, Anil Kumar

Methodology:

This research introduces a deep learning framework for real-time sign language gesture recognition. The system utilizes convolutional neural networks (CNNs) to classify gestures from video input, achieving high accuracy with low latency in processing. The paper discusses various CNN architectures and their effectiveness in recognizing complex hand movements, along with a comparative analysis of model

performance on different datasets. The findings highlight the potential of AI-driven sign language translation systems in real-world applications [3].

4. Hand Gesture Recognition Using MediaPipe and Machine Learning for Sign Language Interpretation

Authors: Sandeep Reddy, Ananya Gupta, Vikram Sharma

Methodology:

The paper presents a hand gesture recognition system leveraging Google's MediaPipe framework. By combining pose estimation and deep learning models, the system accurately translates hand movements into textual representation, enabling real-time sign language interpretation. The study evaluates the performance of MediaPipe-based sign detection on various sign language datasets, emphasizing the advantages of lightweight models for mobile and embedded applications. The researchers propose enhancements such as multi-angle hand tracking and gesture sequence recognition for improved usability.

This literature survey highlights the advancements in sign language recognition systems, ranging from hardware-assisted methods to deep learning-based real-time processing. The studies demonstrate the potential of AI and IoT-based solutions in bridging the communication gap for hearing and speech-impaired individuals, paving the way for more accessible and scalable assistive technologies. The integration of sensors, deep learning, and NLP has significantly improved gesture recognition accuracy, but challenges such as dataset limitations and environmental dependencies remain areas for future research [4].

5. Sign Language Recognition and Translation with Kinect

Authors: Xiujuan Chai, Guang Li, Zhihao Xu, Y. B. Tang

Methodology:

This paper presents an approach utilizing Microsoft's Kinect sensor for sign language recognition and translation. The proposed system captures sign gestures through depth-sensing cameras and applies machine learning models to interpret gestures into spoken language text. The research emphasizes the role of motion tracking technology in improving sign language recognition accuracy and introduces an adaptive algorithm that reduces recognition errors in varying lighting conditions [5].

6. British Sign Language Recognition via Late Fusion of Computer Vision and Leap Motion with Transfer Learning to American Sign Language

Authors: Jordan J. Bird, Aniko Ekart, Diego R. Faria

Methodology:

This study investigates the fusion of computer vision and Leap Motion sensor data to recognize British Sign Language. It further applies transfer learning techniques to adapt the model for American Sign Language, demonstrating the flexibility and potential of AI-driven sign recognition systems. The proposed system employs a multi-modal approach that combines RGB camera input with infrared-based hand tracking, resulting in improved gesture detection accuracy. The authors discuss limitations related to environmental noise and propose future research directions to enhance robustness [6].

7. Machine Translation between Spoken Languages and Signed Languages Represented in Sign Writing

Author: Zifan Jiang

Methodology:

The study examines the use of Sign Writing as an intermediary representation for machine translation between spoken and signed languages. The research highlights the challenges in developing efficient translation systems that facilitate seamless communication between different language users. The proposed method integrates Natural Language Processing (NLP) techniques to automatically convert spoken language text into structured Sign Writing symbols, enabling better cross-linguistic sign communication [7].

SYSTEM DIAGRAMS

3. SYSTEM DIAGRAMS

3.1 SYSTEM ARCHITECTURE

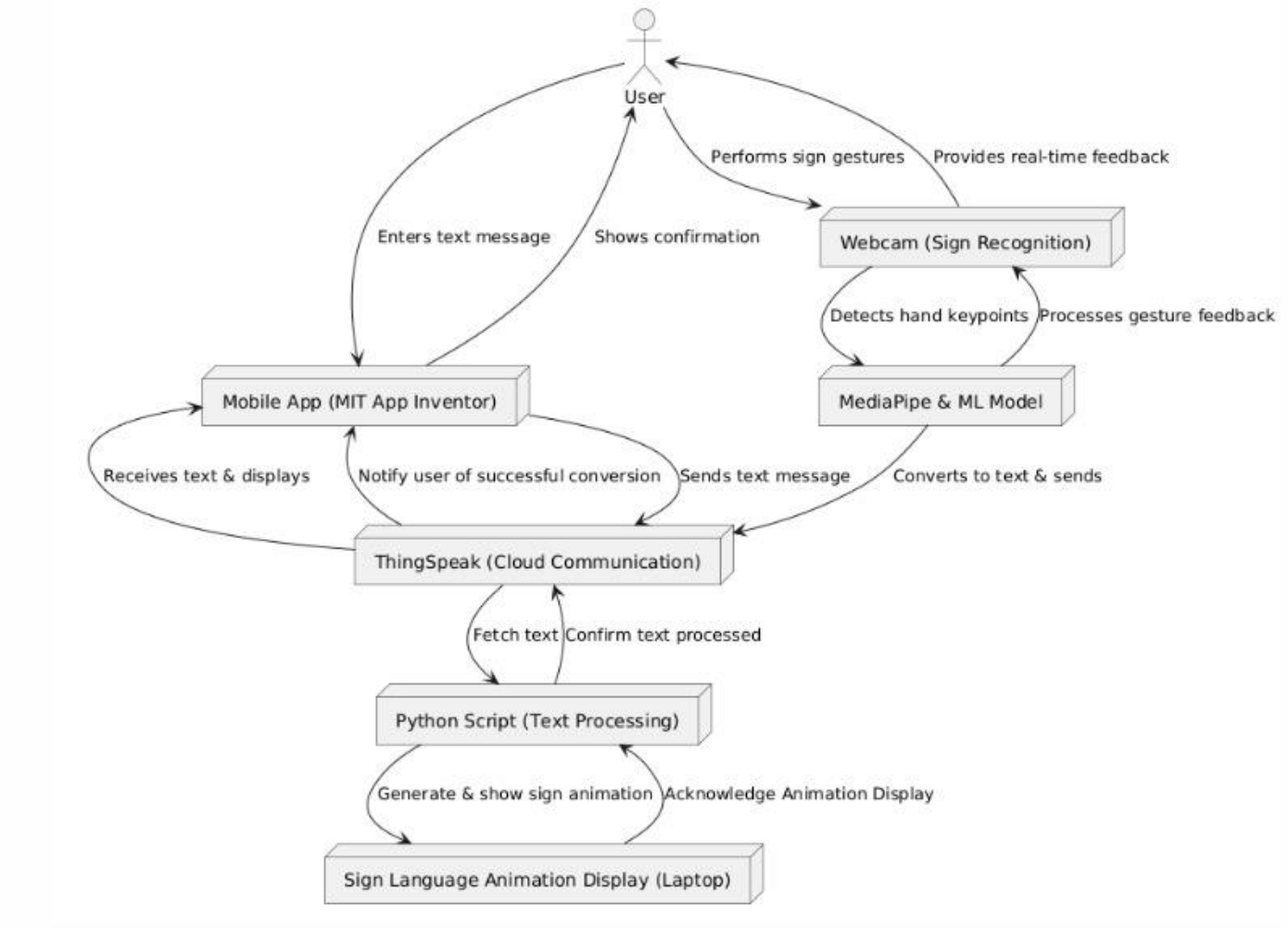

FIGURE 3.1: SYSTEM ARCHITECTURE

3.2 UML DIAGRAMS

3.2.1 USE CASE DIAGRAMS

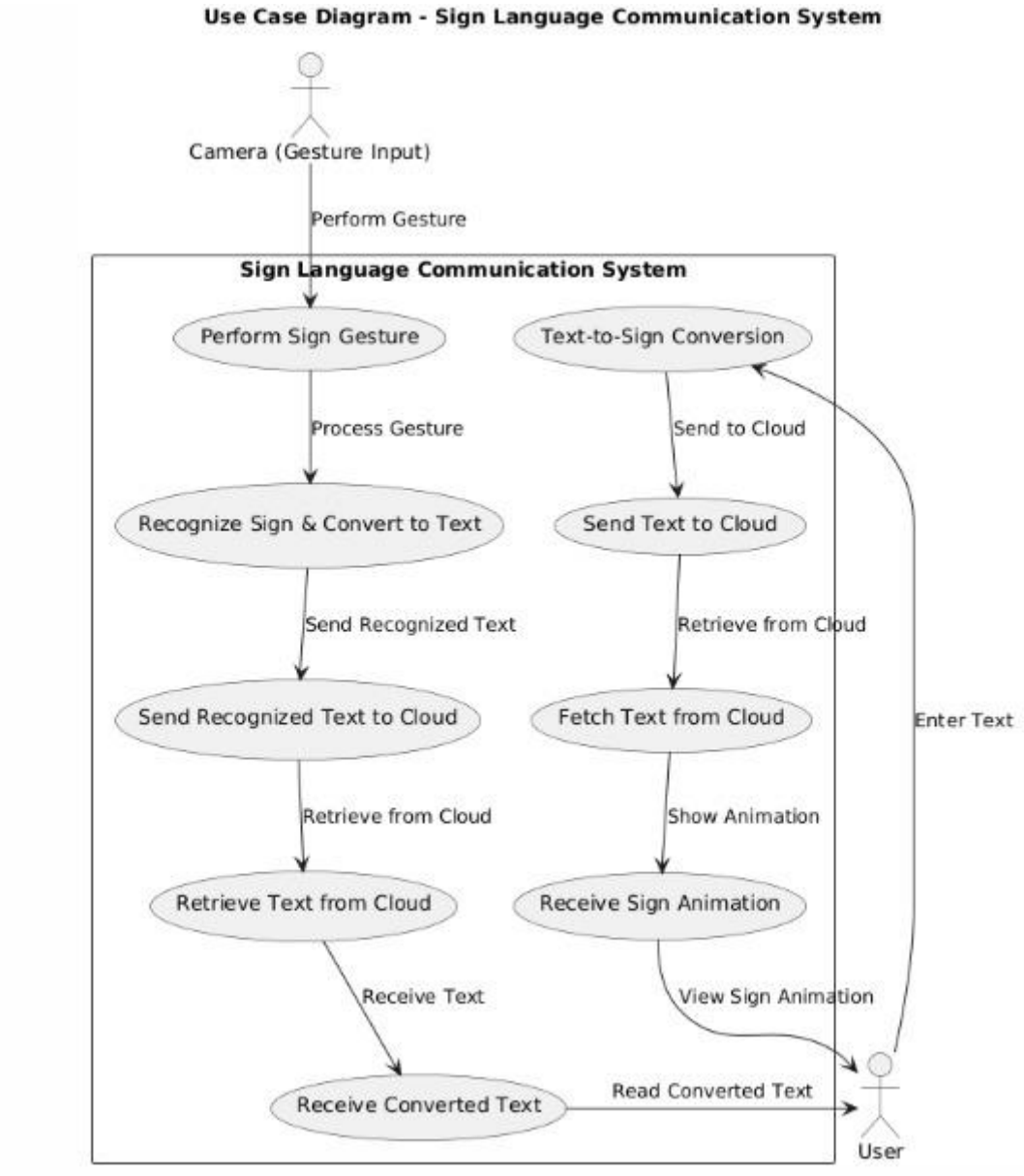

FIGURE 3.2.1: USE CASE DIAGRAM

3.2.2 CLASS DIAGRAM

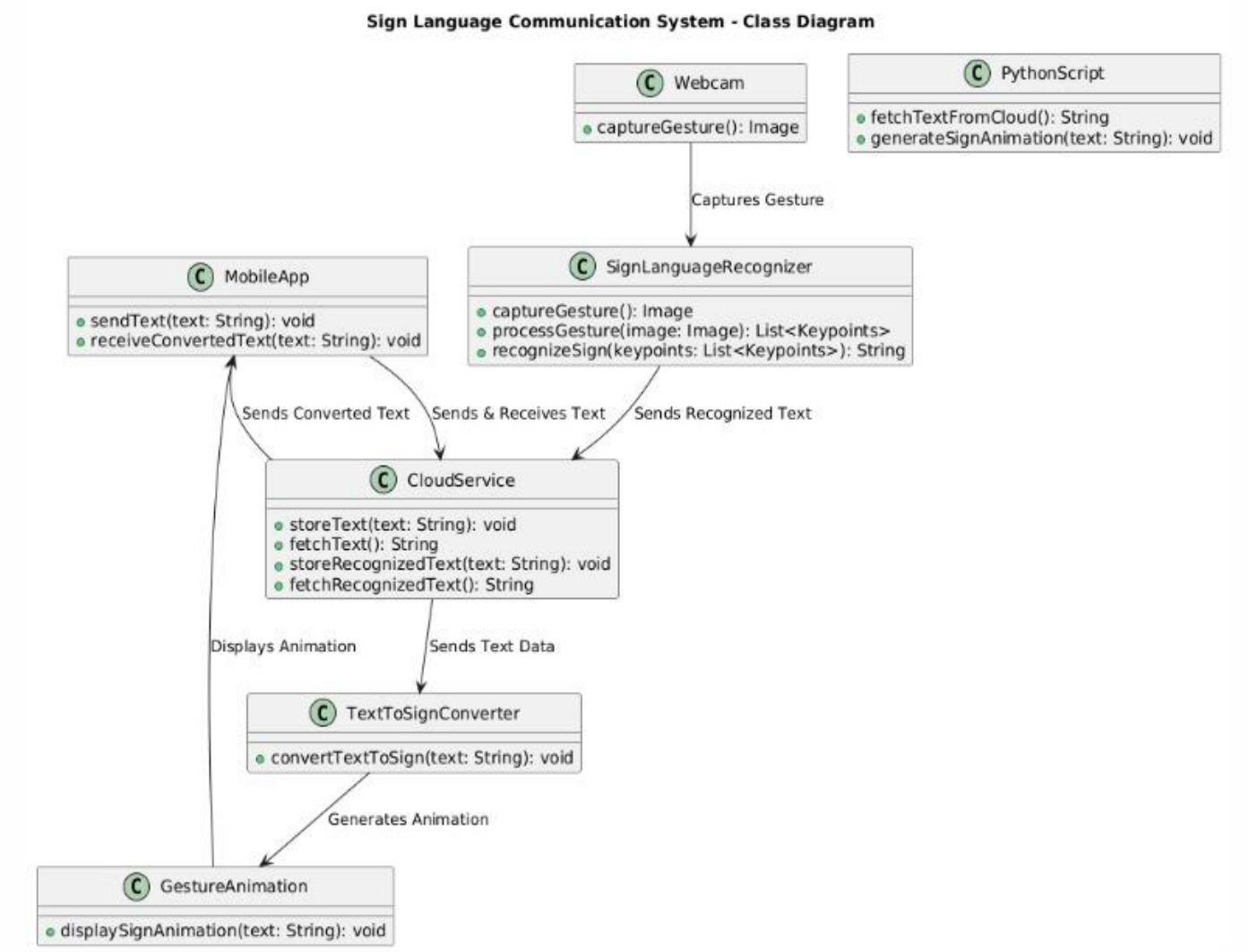

FIGURE 3.2.2: CLASS DIAGRAM

3.2.3 SEQUENCE DIAGRAM

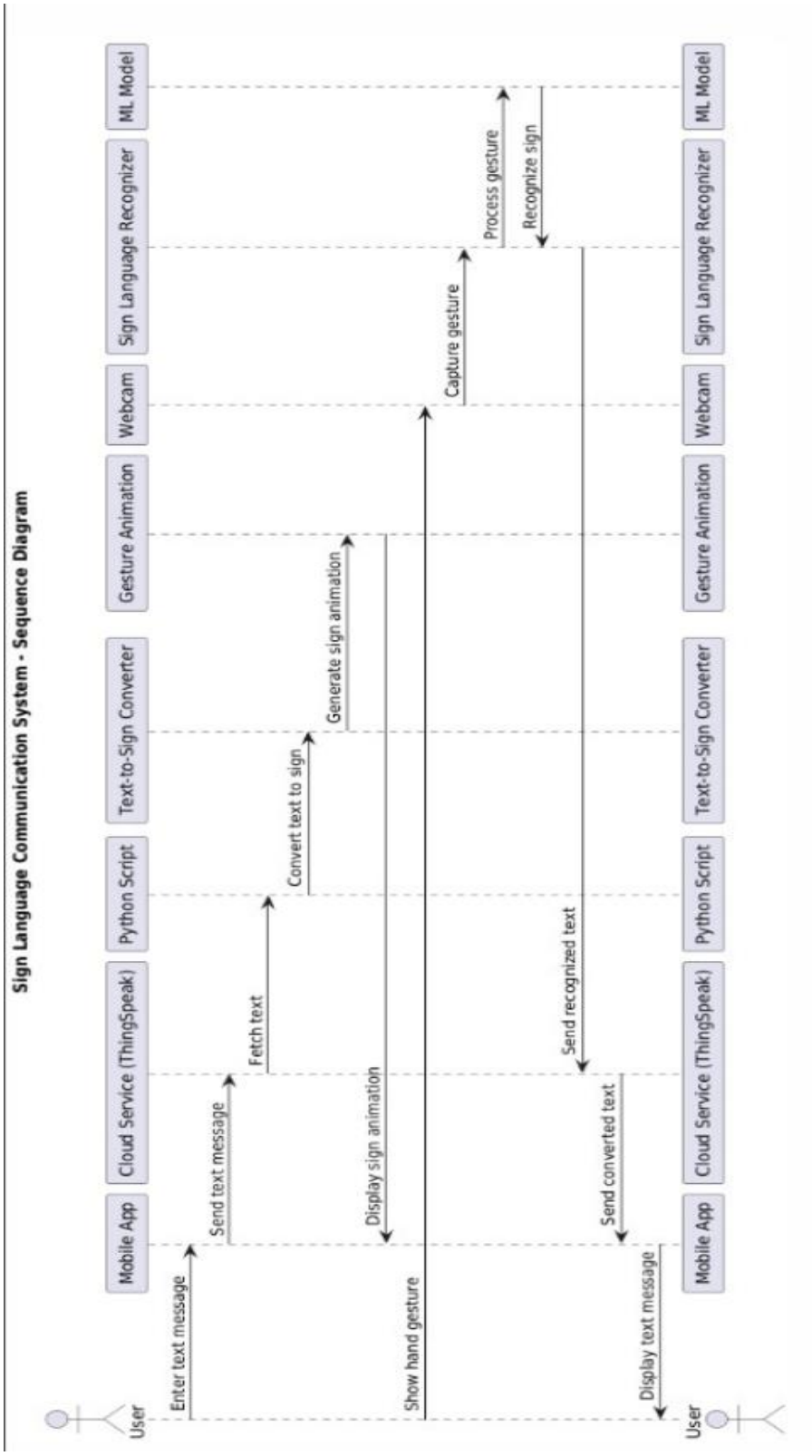

FIGURE 3.2.3: SEQUENCE DIAGRAM

3.2.4 COLLABORATION DIAGRAM

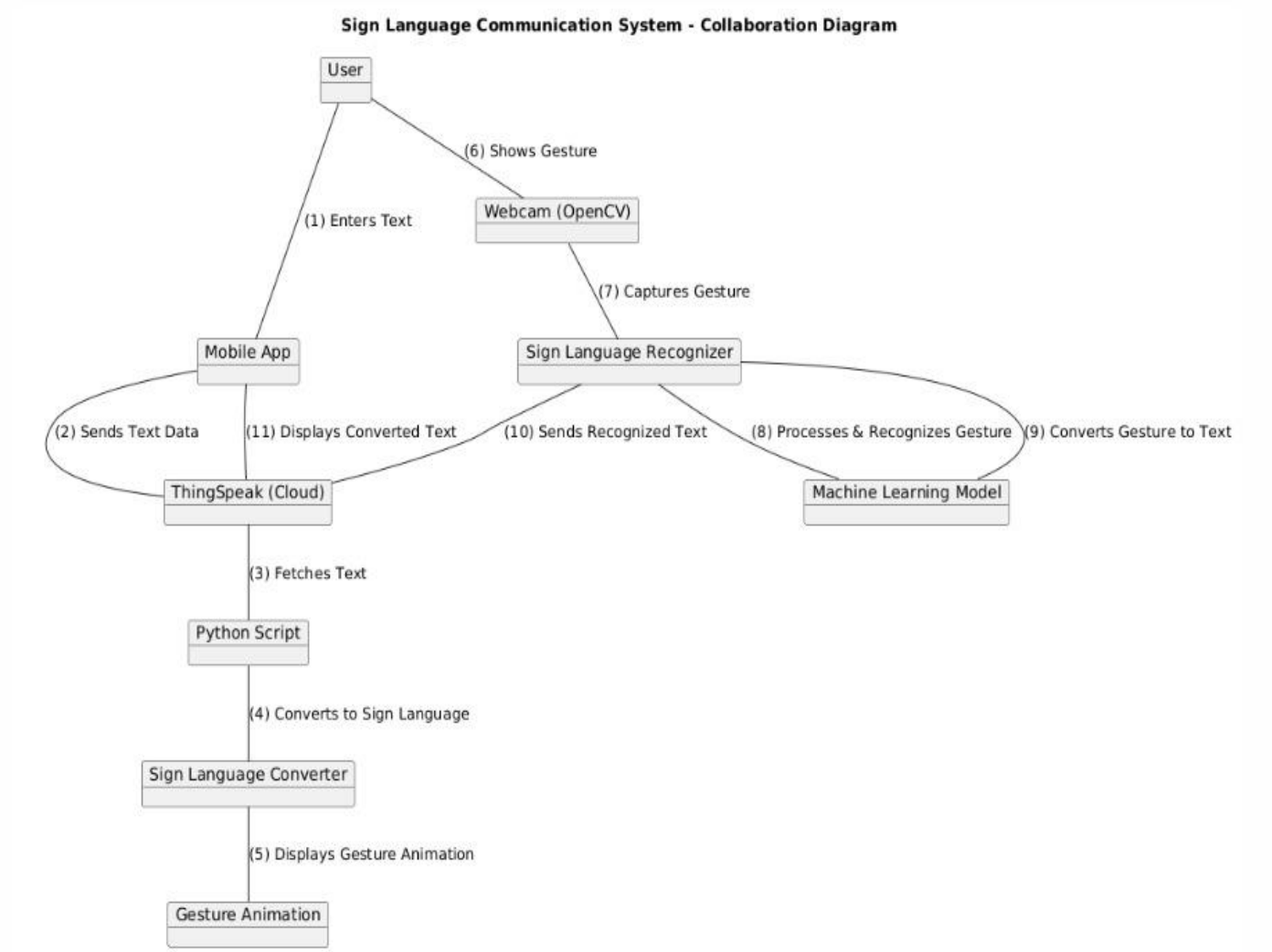

FIGURE 3.2.4: COLLABORATION DIAGRAM

3.2.5 ACTIVITY DIAGRAM

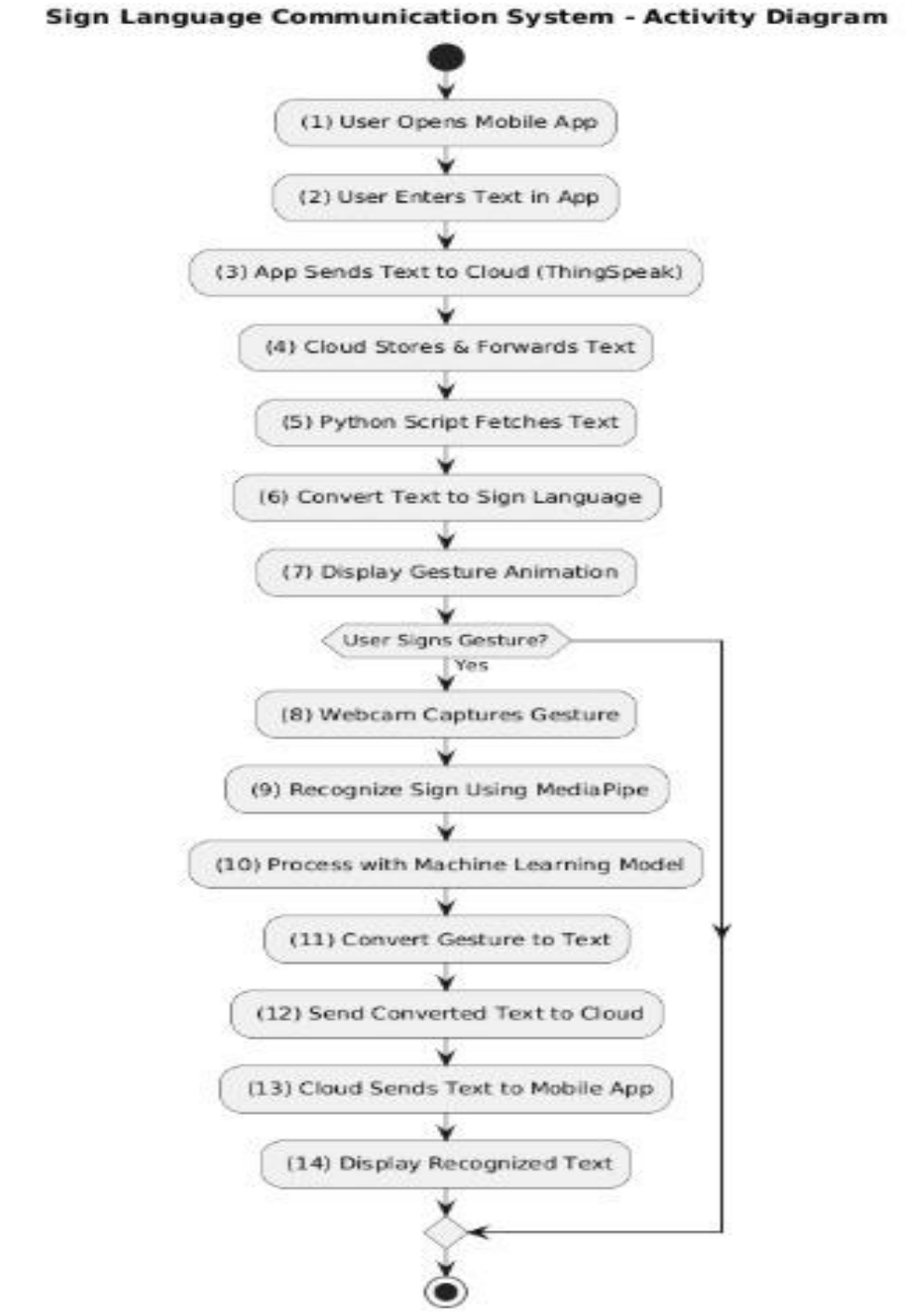

FIGURE 3.2.5: ACTIVITY DIAGRAM

IMPLEMENTATION

4. IMPLEMENTATION

4.1 Modules:

1. Text-to-Sign Language Conversion
2. Sign Language Recognition and Text Conversion
3. Mobile App Development
4. Cloud Communication (ThingSpeak Integration)

Module 1: Text-to-Sign Language Conversion

- Functionality: Converts text messages to sign language gestures.
- Technology Used: Python, ThingSpeak, MIT App Inventor.
- Process:
    1. User enters text in the mobile app.
    2. Python script fetches text from ThingSpeak.
    3. Sign language animation is generated and displayed on a laptop screen.

Module 2: Sign Language Recognition and Text Conversion

- Functionality: Recognizes sign language gestures and converts them to text.
- Technology Used: MediaPipe, Machine Learning, Python.
- Process:
    1. The camera captures real-time sign gestures.
    2. MediaPipe detects key points of the hand movement.
    3. ML model classifies gestures and converts them to text.
    4. Text is transmitted to the mobile app via ThingSpeak.

Module 3: Mobile App Development

- Functionality: Provides a user-friendly interface for communication.
- Technology Used: MIT App Inventor.
- Process:
    1. Allows users to enter and receive text messages.
    2. Displays received text converted from sign language.

Module 4: Cloud Communication (ThingSpeak Integration)

- Functionality: Facilitates real-time data exchange between mobile and laptop.
- Technology Used: ThingSpeak (IoT platform).
- Process:
    1. Text messages from the app are sent to ThingSpeak.
    2. Python script fetches the text for gesture animation.
    3. Recognized sign language text is transmitted back to the mobile app.

4.2 Design Description

The proposed Sign Language Translation System is structured to ensure seamless two-way communication between hearing and speaking-impaired individuals and others. The system workflow is divided into two primary processes: Mobile App to Sign Conversion and Sign to Mobile App Communication.

4.2.1 System Workflow

Mobile App to Sign Conversion:

- Input: The user types a text message in the mobile application, which is designed using MIT App Inventor. This app serves as the communication medium for individuals who do not understand sign language but wish to communicate with someone who does.
- Processing: The text message is transmitted to a Python script running on a laptop through the IoT platform ThingSpeak. Upon receiving the message, the Python script processes the text and maps it to corresponding sign language gestures using an animation module.
- Output: The laptop screen displays the animated sign language gestures corresponding to the received message, allowing the hearing-impaired individual to visually interpret the information.

Sign to Mobile App Communication:

- Input: The hearing-impaired user performs sign language gestures in front of a laptop camera. The camera captures the real-time video of the gestures.
- Processing: The system utilizes the MediaPipe framework to detect key points on the hands, fingers, and relevant facial expressions. A pre-trained machine learning model processes these detected key points and maps them to corresponding text representations.
- Output: The interpreted text is sent to the mobile application via ThingSpeak. The receiving user can then read the converted text on the app and respond accordingly, ensuring smooth, real-time communication.

4.2.2 System Components

- Mobile Application: Developed using MIT App Inventor, it serves as the user interface for sending and receiving messages.
- ThingSpeak: Acts as the IoT communication bridge between the mobile application and the laptop.
- Python Script: Handles text processing, sign language animation, and gesture recognition.
- MediaPipe Framework: Facilitates real-time hand and facial key point detection for sign language recognition.
- Machine Learning Model: Maps gestures to text, enabling accurate translation of sign language into readable text.
- Laptop with Camera: Captures sign language gestures and processes text-to-sign conversions for display.

This structured workflow ensures real-time, cost-effective, and accessible communication between hearing-impaired and non-impaired individuals while integrating cutting-edge technology to enhance inclusivity.

Front-End:

- The mobile application is developed using MIT App Inventor, designed with an intuitive and user-friendly interface.
- Users can input text messages into the app, which are then processed for sign language conversion.
- The mobile app also displays text output that has been translated from sign language gestures captured via the laptop's camera.
- The interface includes clear navigation buttons for sending and receiving messages, ensuring accessibility for all users.

Back-End:

- The core processing is handled by a Python script that manages both text-to-sign and sign-to-text conversions.
- For text-to-sign conversion, the Python script retrieves messages from the mobile app via ThingSpeak and processes them into animated sign language gestures.
- For sign-to-text conversion, the system uses MediaPipe to detect keypoints from real-time video input, and a machine learning model interprets the gestures into text.
- The interpreted text is transmitted back to the mobile app through ThingSpeak for user consumption.
- IoT integration ensures seamless communication between the mobile app and the laptop, allowing real-time processing of messages and gestures.

- The architecture is designed for scalability and cost-effectiveness, enabling widespread deployment without requiring expensive hardware.

4.3 ADVANTAGES & APPLICATIONS

Advantages

1. Bi-Directional Communication: Supports both text-to-sign and sign-to-text conversion.
2. Real-Time Processing: Ensures quick response and user satisfaction.
3. Accessible and Scalable: Cost-effective and easily deployable for large-scale use.
4. Empowers Inclusivity: Enhances communication opportunities for disabled individuals.

Applications

1. Education: Facilitates teaching and learning for hearing and speaking-impaired students.
2. Healthcare: Enhances patient-doctor communication for individuals with impairments.
3. Customer Service: Provides accessible interaction in public service sectors.
4. Personal Communication: Improves daily interactions for disabled individuals.

The Sign Language Conversion System offers a cost-effective, scalable, and real-time communication solution for hearing and speaking-impaired individuals. By integrating machine learning, IoT, and real-time processing, this system enhances accessibility and inclusivity, ensuring that communication barriers are significantly reduced.

SYSTEM REQUIREMENTS

5. SYSTEM REQUIREMENTS

5.1 Hardware Requirements

1. Laptop/PC with a camera.
2. Mobile device with MIT App Inventor app.

5.2 Software Requirements

1. Python (with MediaPipe and ML libraries).
2. MIT App Inventor for mobile app development.
3. ThingSpeak for cloud-based IoT communication.
4. Machine Learning model for sign gesture recognition.

5.3 Software Description:

5.3.1 Python

Python is one of those rare languages which can claim to be both simple and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Features of Python

- Simple
- Easy to learn
- Free and open source
- High-level language
- Portable
- Interpreted
- Object Oriented
- Extensible

5.3.2 Machine Learning

Machine learning (ML) is a type of artificial intelligence (AI) that allows computers to learn from data and perform tasks without needing explicit programming. It uses algorithms to analyse data, identify patterns, and make decisions.

How does ML work?

- ML uses neural networks and deep learning to learn and improve from data.
- ML algorithms improve over time as they are exposed to more data.
- ML models are the output of the program after running an algorithm on training data.

Uses of ML

- Personalization
- Data analysis
- Healthcare
- Manufacturing
- Financial services
- Retail

5.3.3 Modules Used in Project:

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code

- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupiter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

5.4 Testing Products:

5.4.1 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

- Field testing will be performed manually and functional tests will be written in detail.

Test objectives:

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested:

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

5.4.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

5.4.3 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Testing is a crucial phase in the development of the Sign Language Conversion System to ensure its reliability, accuracy, and performance. Various types of testing are implemented to validate different components of the system.

Test case ID	**Description**	**Input**	**Expected output**	**Result**
TC001	Verify message transmission from mobile app	Text message	Message received by ThingSpeak	✓ YES/NO
TC002	Check text to sign conversion	"Hello"	Corresponding sign animation	✓ YES/NO

TC003	Validate sign recognition accuracy	Sign gesture for "Thank You"	"Thank You" displayed as text	✓ YES/NO
TC004	Test latency in sign interpretation	Real-time gesture	Response time < 2 sec	✓ YES/NO
TC005	Ensure UI usability for new users	User navigation	Easy interaction, no confusion	✓ YES/NO

TABLE 5.4.4: TEST RESULTS

SAMPLE CODING

6. SAMPLE CODING

```
import os

import cv2

import numpy as np

import mediapipe as mp

import tensorflow as tf

from tensorflow.keras.models import load_model

import time

import urllib.request

import json

import string

import threading

from PIL import Image

from collections import deque


# ThingSpeak Channel Settings

THINGSPEAK_API_KEY = "IP6U7YPZJW9WNANU"  # Replace with your API key

THINGSPEAK_URL =
f"https://api.thingspeak.com/channels/466832/feeds/last.json?api_key={THINGSPEAK_API_KEY}"


# List of predefined words that have corresponding GIFs

isl_gif = [

    'any questions', 'are you angry', 'are you busy', 'are you hungry', 'are you sick', 'be careful',

    'can we meet tomorrow', 'did you book tickets', 'did you finish homework', 'do you go to office',

    'do you have money', 'do you want something to drink', 'do you want tea or coffee', 'do you watch TV',

    'dont worry', 'flower is beautiful', 'good afternoon', 'good evening', 'good morning', 'good night',
```

```
    'good question', 'had your lunch', 'happy journey', 'hello what is your name', 'how many people are there in your family',
    'i am a clerk', 'i am bore doing nothing', 'i am fine', 'i am sorry', 'i am thinking', 'i am tired',
    'i dont understand anything', 'i go to a theatre', 'i love to shop', 'i had to say something but i forgot',
    'i have headache', 'i like pink colour', 'i live in nagpur', 'lets go for lunch', 'my mother is a homemaker',
    'my name is john', 'nice to meet you', 'no smoking please', 'open the door', 'please call me later',
    'please clean the room', 'please give me your pen', 'please use dustbin dont throw garbage',
    'please wait for sometime', 'shall I help you', 'shall we go together tomorrow', 'sign language interpreter',
    'sit down', 'stand up', 'take care', 'there was traffic jam', 'wait I am thinking', 'what are you doing',
    'what is the problem', 'what is todays date', 'what is your father do', 'what is your job',
    'what is your mobile number', 'what is your name', 'whats up', 'when is your interview', 'when we will go',
    'where do you stay', 'where is the bathroom', 'where is the police station', 'you are wrong'
]
# Alphabet list
arr = list(string.ascii_lowercase)

# Global variables for threading
latest_text = ""
gif_frame_queue = deque()  # Queue to store GIF frames
current_gif_frame = None
gif_playing = False
def fetch_text_from_thingspeak():
    """Fetches the latest text from ThingSpeak"""
    global latest_text
    while True:
        try:
```

```
            response = urllib.request.urlopen(THINGSPEAK_URL)
            data = json.load(response)
            text = data["field1"]  # Assuming text is stored in 'field1'
            latest_text = text.strip().lower()
            print("Fetched text from ThingSpeak:", latest_text)  # Debug print
        except Exception as e:
            print(f"Error fetching data from ThingSpeak: {e}")
        time.sleep(2)  # Fetch every 2 seconds
def load_gif_frames(gif_path):
    """Loads all frames of a GIF into a queue"""
    global gif_frame_queue, gif_playing
    gif = Image.open(gif_path)
    gif_frame_queue.clear()
    try:

        while True:
            frame = np.array(gif.convert('RGB'))
            gif_frame_queue.append(frame)
            gif.seek(len(gif_frame_queue))  # Move to the next frame
    except EOFError:
        pass  # End of GIF
    gif_playing = True
def display_gif_or_letters():
    """Displays GIF or letters based on the latest text"""
    global latest_text, gif_frame_queue, current_gif_frame, gif_playing
    while True:
```

```
if latest_text:
    text = latest_text
    for c in string.punctuation:
        text = text.replace(c, "")
    print("Processing text:", text)  # Debug print


    if text in isl_gif:
        print("Text matches ISL GIF:", text)  # Debug print
        gif_path = f'ISL_Gifs/{text}.gif'
        if os.path.exists(gif_path):
            threading.Thread(target=load_gif_frames, args=(gif_path,)).start()
        else:
            print("GIF not found for:", text)
            gif_playing = False
    else:
        print("Text does not match ISL GIF. Displaying as letters.")  # Debug print
        gif_playing = False
        gif_frame_queue.clear()
        current_gif_frame = None
        for char in text:
            if char in arr:
                image_address = f'letters/{char}.jpg'
                if os.path.exists(image_address):
                    image = cv2.imread(image_address)
                    if current_gif_frame is None:
                        current_gif_frame = image
```

```
                    else:
                        current_gif_frame = np.hstack((current_gif_frame, image))
                else:
                    print("Letter image not found for:", char)
        time.sleep(1)  # Check for updates every second

# Initialize mediapipe
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
mpDraw = mp.solutions.drawing_utils

# Load the gesture recognition model
model = load_model('model.hdf5')

# Load gesture names
with open('gesture.names8', 'r') as f:
    classNames = f.read().split('\n')
print(classNames)

# Initialize the webcam
cap = cv2.VideoCapture(0)

# Start threads for ThingSpeak and GIF/letter display
threading.Thread(target=fetch_text_from_thingspeak, daemon=True).start()
threading.Thread(target=display_gif_or_letters, daemon=True).start()
```

```
while True:
    # Read each frame from the webcam
    _, frame = cap.read()
    image_width, image_height = frame.shape[1], frame.shape[0]
    frame = cv2.flip(frame, 1)

    # Get hand landmark prediction
    result = hands.process(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    className = ''

    if result.multi_hand_landmarks:
        landmarks = []
        for handslms in result.multi_hand_landmarks:
            for i, lm in enumerate(handslms.landmark):
                lmx = int(lm.x * image_width)
                lmy = int(lm.y * image_height)
                if i == 0:
                    temp_x = lmx
                    temp_y = lmy
                    lmx = 0
                    lmy = 0
                else:
                    lmx = lmx - temp_x
                    lmy = lmy - temp_y
                landmarks.append(lmx)
```

```
            landmarks.append(lmy)


        mpDraw.draw_landmarks(frame, handslms, mpHands.HAND_CONNECTIONS)
        landmarks = np.array(landmarks)
        landmarks = landmarks / abs(max(landmarks, key=abs))
        landmarks = landmarks.reshape(42, 1)
        prediction = model.predict(np.array([landmarks]))
        classID = np.argmax(prediction)
        className = classNames[classID]


        # Display emoji overlay
        emoji_path = f"emojis/{classID}.png"
        if os.path.exists(emoji_path):
            overlay = cv2.imread(emoji_path, cv2.IMREAD_UNCHANGED)
            if overlay.shape[2] == 4:  # Check if the image has an alpha channel
                overlay = cv2.resize(overlay, (100, 100))
                x_offset, y_offset = 10, 10
                for c in range(0, 3):
                    frame[y_offset:y_offset + overlay.shape[0], x_offset:x_offset + overlay.shape[1], c] = \
                        overlay[:, :, c] * (overlay[:, :, 3] / 255.0) + \
                        frame[y_offset:y_offset + overlay.shape[0], x_offset:x_offset + overlay.shape[1], c] * (1.0
- overlay[:, :, 3] / 255.0)
            else:
                overlay = cv2.resize(overlay, (100, 100))
                x_offset, y_offset = 10, 10
                frame[y_offset:y_offset + overlay.shape[0], x_offset:x_offset + overlay.shape[1]] = overlay
```

```
# Add the recognized gesture text to the frame
    cv2.putText(frame, className, (250, 200), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)

    # Display GIF or letters
    if gif_playing and gif_frame_queue:
        current_gif_frame = gif_frame_queue.popleft()
        gif_frame_queue.append(current_gif_frame)  # Cycle through frames
    elif current_gif_frame is not None:
        pass  # Keep displaying the current frame

    # Resize the GIF or letter frame
    if current_gif_frame is not None:
        gif_frame_resized = cv2.resize(current_gif_frame, (image_width, image_height))
    else:
        gif_frame_resized = np.zeros((image_height, image_width, 3), dtype=np.uint8)

    # Combine the webcam feed and GIF/letter display vertically
    combined_frame = np.hstack((frame, gif_frame_resized))

    # Show the final output
    cv2.imshow("Output", combined_frame)

    if cv2.waitKey(1) == ord('q'):
```

```
        break

# Release the webcam and destroy all active windows

cap.release()

cv2.destroyAllWindows()
```

SCREEN SHOTS

7. SAMPLE SCREENSHOTS

7.1 SIGN LANGUAGE TO TEXT

FIGURE 7.1.1: SIGN TO TEXT

FIGURE 7.1.2: SIGN TO TEXT

7.2 TEXT TO SIGN LANGUAGE

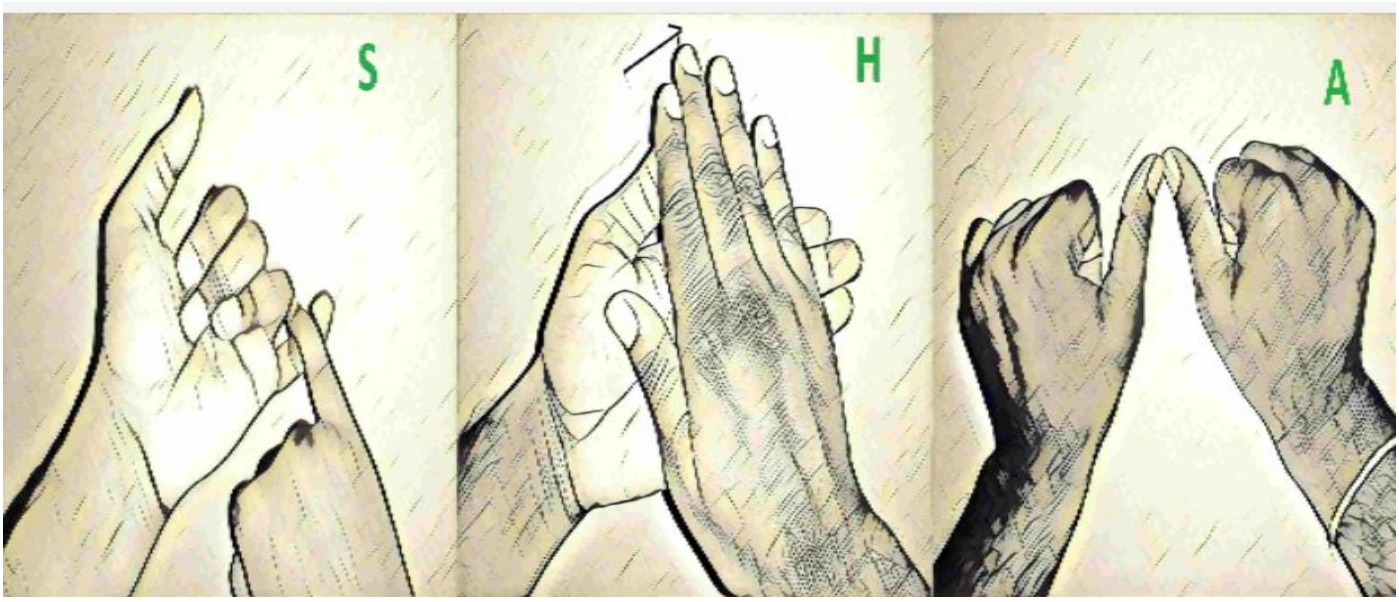

FIGURE 7.2.1: TEXT TO SIGN

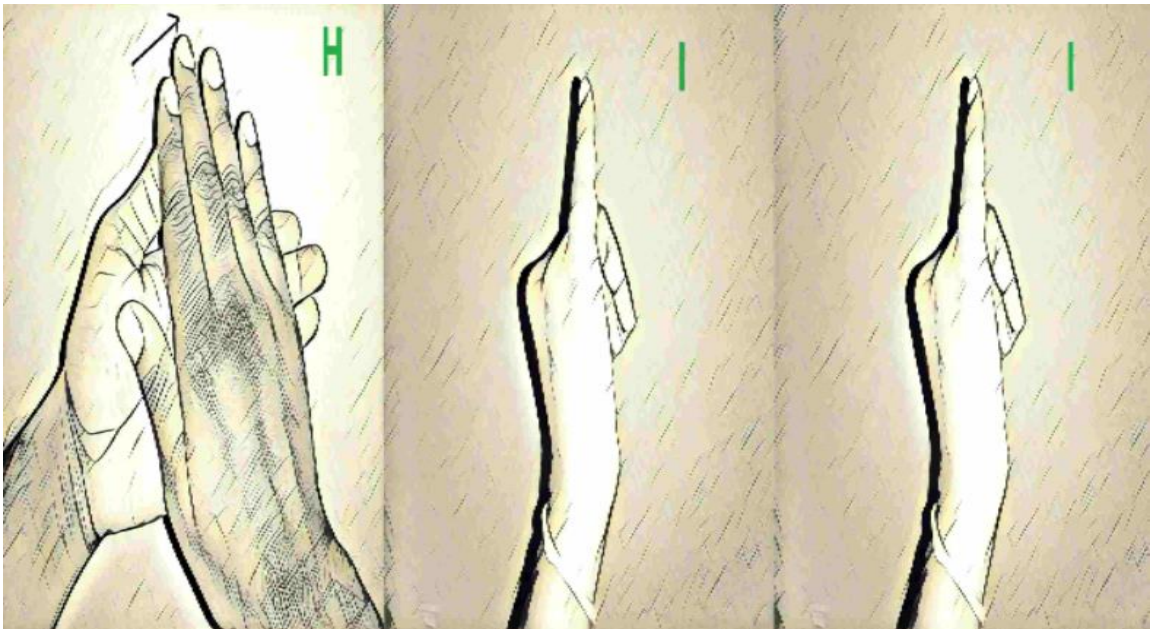

FIGURE 7.2.2: TEXT TO SIGN

7.3 MESSAGE IN MOBILE APP

FIGURE 7.3.1: MESSAGE GIVEN IN MOBILE APP

7.4 COMMANDS IN COMMAND PROMPT

```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nandh\OneDrive\Documents\Desktop\only gesture (2)\only gesture>python final1.py
2025-04-05 21:44:06.614337: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly di
fferent numerical results due to floating-point round-off errors from different computation orders. To turn them off, se
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-04-05 21:44:13.100085: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly di
fferent numerical results due to floating-point round-off errors from different computation orders. To turn them off, se
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1743869667.498593    5440 inference_feedback_manager.cc:114] Feedback manager requires a model with a single
 signature inference. Disabling support for feedback tensors.
2025-04-05 21:44:27.500512: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to
use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow
with the appropriate compiler flags.
W0000 00:00:1743869667.557485   20528 inference_feedback_manager.cc:114] Feedback manager requires a model with a single
 signature inference. Disabling support for feedback tensors.
C:\Users\nandh\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\src\optimizers\base_optimizer.py:86: User
Warning: Argument `decay` is no longer supported and will be ignored.
  warnings.warn(
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be e
mpty until you train or evaluate the model.
WARNING:absl:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initialized
optimizer.
['I want u to stop here ', 'Give me some food', 'I have a question', 'Iam feeling good', 'Its time to take medicine', 'I
am upset', 'I want to go out', 'Can u give me directions ']
Fetched text from ThingSpeak: hii
```

```
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
W0000 00:00:1743869678.847195   15484 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is
 only supported for the square ROI. Provide IMAGE_DIMENSIONS or use PROJECTION_MATRIX.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 367ms/stepProcessing text: hii
Text does not match ISL GIF. Displaying as letters.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 392ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
Fetched text from ThingSpeak: hii
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
```

```
Text does not match ISL GIF. Displaying as letters.
Fetched text from ThingSpeak: hii
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Fetched text from ThingSpeak: hii
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Fetched text from ThingSpeak: hii
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Fetched text from ThingSpeak: hii
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Processing text: hii
Text does not match ISL GIF. Displaying as letters.
Fetched text from ThingSpeak: hii
```

CONCLUSION

8. CONCLUSION

The Sign Language Conversion System successfully addresses the communication challenges faced by hearing and speech-impaired individuals by providing a real-time, bi-directional solution that translates text into animated sign language and sign gestures into text. Using MediaPipe for gesture recognition, machine learning for interpretation, and ThingSpeak for IoT-based communication, the system ensures seamless interaction between users and non-signers.

This project is designed to be cost-effective, scalable, and easy to implement, eliminating reliance on expensive interpreters or specialized hardware. The mobile application developed using MIT App Inventor enhances accessibility, allowing users to communicate effortlessly.

By integrating AI, IoT, and real-time processing, this system can be widely applied in education, healthcare, workplaces, and public services, empowering hearing and speech-impaired individuals with greater independence.

FUTURE ENHANCEMENT

9. FUTURE ENHANCEMENTS

Future advancements in communication technologies for individuals with impairments hold immense promise in creating a more inclusive and accessible world. By incorporating support for multiple sign languages, the system could break down language barriers, offering users the ability to communicate in their native sign language, whether it's American Sign Language (ASL), British Sign Language (BSL), or others. This would enhance global communication, ensuring that individuals can interact seamlessly across different cultures.

The development of improved gesture recognition accuracy would enable the system to understand a wider range of complex, fast, or nuanced gestures, even in diverse environments. With more precise recognition, the technology could better cater to the varied needs of users, improving the overall communication experience.

Integrating speech-to-sign technology would bridge the gap between spoken and signed communication, facilitating real-time translation from speech into sign language. This would foster more inclusive interactions in everyday conversations, education, and professional settings.

REFERENCES

10. REFERENCES

1. Boris Mocialov, Graham Turner, Katrin Lohan, Helen Hastie (2017) Towards Continuous Sign Language Recognition with Deep Learning. Journal Proc. of the Workshop on the Creating Meaning With Robot Assistants: The Gap Left by Smart Devices

2. DL Jaffe (1994) Evolution of Mechanical Fingerspelling Hands for People Who Are Deaf-Blind Aug, 31(3):236-44, NIH.

3. Rajesh Kumar, Pooja Sharma, Anil Kumar, A Deep Learning-Based Approach for Real-Time Sign Language Gesture Recognition, Ijraset Journal For Research in Applied Science and Engineering Technology 2024, 2321-9653.

4. Sandeep Reddy, Ananya Gupta, Vikram Sharma (2023) Hand Gesture Recognition Using MediaPipe and Machine Learning for Sign Language Interpretation, MDPI, *59*(1), 96

5. Xiujuan Chai, Guang Li, Zhihao Xu, Y. B. Tang (2015) Sign Language Recognition and Translation with Kinect, August 2015, International Journal of Applied Engineering Research

6. Jordan J. Bird, Aniko Ekart, Diego R. Faria (2020) British Sign Language Recognition via Late Fusion of Computer Vision and Leap Motion with Transfer Learning to American Sign Language, MDPI, *20*(18), 5151.

7. Zifan Jiang, Amit Moryossef, Mathias Müller, Sarah Ebling (2023) Machine Translation between Spoken Languages and Signed Languages Represented in Sign Writing, findings-eacl.127, Association for Computational Linguistics

8. Starner, T., Weaver, J., & Pentland, A. (1998). Real-time American Sign Language recognition using desk and wearable computer-based video. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12), 1371-1375.

9. Cooper, H., Holt, B., & Bowden, R. (2011). Sign language recognition. In Visual Analysis of Humans (pp. 539-562). Springer.

10. Ong, S. C., & Ranganath, S. (2005). Automatic sign language analysis: A survey and the future beyond lexical meaning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(6), 873-891.

11. Koller, O., Zargaran, S., Ney, H., & Bowden, R. (2016). Deep sign: Hybrid CNN-HMM for continuous sign language recognition. Proceedings of the British Machine Vision Conference.

12. Camgoz, N. C., Hadfield, S., Koller, O., & Bowden, R. (2017). SubUNets: End-to-end hand shape and continuous sign language recognition. Proceedings of the IEEE International Conference on Computer Vision.

13. Pigou, L., Dieleman, S., Kindermans, P. J., & Schrauwen, B. (2015). Sign language recognition using convolutional neural networks. European Conference on Computer Vision (ECCV).

14. Cui, R., Liu, H., & Zhang, C. (2017). Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. IEEE Transactions on Image Processing, 26(7), 3333-3347.

15. Shi, X., Zhao, Z., Chen, X., & Wang, L. (2019). Continuous sign language recognition with sequence-based convolutional networks. IEEE Transactions on Multimedia, 21(7), 1892-1903.

16. Li, D., Rodriguez, C., Yu, X., & Li, H. (2020). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision.

17. Alon, J., Athitsos, V., Yuan, Q., & Sclaroff, S. (2009). A unified framework for gesture recognition and spatiotemporal gesture segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(9), 1685-1699.

18. Kumar, R., Garg, R., & Meena, Y. K. (2022). A real-time Indian Sign Language recognition system using deep learning. International Journal of Advanced Computer Science and Applications, 13(3), 220-227.

19. Ghosh, A., Das, N., & Nasipuri, M. (2019). A deep learning framework for recognition of sign language gestures captured with leap motion. Neural Computing and Applications, 31(6), 1865-1875.

20. Murthy, G. R., & Jadon, R. S. (2009). A review of vision-based hand gestures recognition. International Journal of Information Technology and Knowledge Management, 2(2), 405-410.

21. Fang, G., Gao, W., & Zhao, D. (2004). Large vocabulary sign language recognition based on fuzzy decision trees. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 34(3), 305-314.

22. Kadous, M. W. (1996). Machine recognition of Auslan signs using power gloves: Towards large-lexicon recognition of sign language. Proceedings of the Workshop on the Integration of Gesture in Language and Speech.

23. Liang, R. H., & Ouhyoung, M. (1998). A real-time continuous gesture recognition system for sign language. Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition.

24. Bauer, B., & Kraiss, K. F. (2001). Video-based sign recognition using self-organizing maps. IEEE International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems.

25. Zhang, D., & Zhou, Z. (2013). Recognizing signs from videos with pyramid histogram of oriented gradients. IEEE Transactions on Circuits and Systems for Video Technology, 23(6), 1072-1080.

26. Almasre, E. A., & Elaraby, S. (2021). Hybrid deep learning model for sign language recognition. International Journal of Computer Applications, 183(3), 35-41.

27. Mohandes, M., Deriche, M., & Aliyu, S. (2014). Arabic sign language recognition using deep learning and computer vision. Artificial Intelligence Review, 42(3), 647-665.

28. P. Vijayalakshmi, M. Aarthi, Sign language to speech conversion, (2016) International Conference on Recent Trends in Information Technology (ICRTIT), IEEE.

29. Kohsheen Tiku, Jayshree Maloo, Aishwarya Ramesh, (2020). Real-time Conversion of Sign Language to Text and Speech, IEEE Xplore

30. T. Hanke, HamNoSys (2004) Representing Sign Language Data in Language Resources and Language Processing Contexts, University of Hamburg, Binderstrabe 34, 20146 Hamburg, Germany.

31. S. Prillwitz, R. Leven, H. Zienert, T. Hanke and J. Henning, (1989) HamNoSys Version 2.0: Hamburg Notation System for Sign Languages: An Introductory Guide, International Studies on Sign Language and Communication of the Deaf, Signum Press, Hamburg, Germany, vol. 5.

32. R. Kennaway, (2001) Synthetic Animation of Deaf Signing Gestures, In International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction, London, UK, Springer, pp. 149-174, (2001).