SIGN LANGUAGE INTERPRETER FOR HEARING IMPAIRED

A PROJECT REPORT

Submitted by

ARAVINDAKUMAR G	(622221104005)
GOKUL K	(622221104016)
SAILESH R	(622221104039)
SUHASINI L	(622221104055)

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PAVAI COLLEGE OF TECHNOLOGY

PACHAL, NAMAKKAL - 637018

ANNA UNIVERSITY: CHENNAI 600 025

MAY - 2025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“Sign Language Interpreter for Hearing Impaired”** is the Bonafide work of **ARAVINDAKUMAR G (622221104005), GOKUL K (622121104016), SAILESH R (622221104039), SUHASINI L (622221104055)** who have carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Mr. M. SUNDARAM, M.E.,	**Mr. M. SUNDARAM, M.E.,**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Computer Science and Engineering	Department of Computer Science and Engineering
Pavai College of Technology,	Pavai College of Technology,
Namakkal-637018	Namakkal-637018

Submitted for the university project viva-voce held on………………..........

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We **ARAVINDAKUMAR G, GOKUL K, SAILESH R** and **SUHASINI L** hereby declare that the project report titled done by us under the guidance of **Mr. M. SUNDARAM, M.E.,** at **PAVAI COLLEGE OF TECHNOLOGY, NAMAKKAL** is submitted in partial fulfilment of the requirements for the award of **BACHELOR OF ENGINEERING** degree in **COMPUTER SCIENCE AND ENGINEERING**. Certified further that, to the best of my knowledge, the work reported here in does not form part of any other project report or dissertation on the basic of which a degree or award was conferred on an earlier occasion on this or any other candidate.

1.

2.

3.

4.

DATE:

PLACE: NAMAKKAL **SIGNATURE OF THE CANDIDATES**

ACKNOWLEDGEMENTS

A great deal of arduous work and effort has been spent in implementing this project work. Several special people have guided us and have contributed significantly to this work and so this becomes obligatory to record our thanks to them.

We express our profound gratitude to our beloved chairman **Shri. CA.N.V. NATARAJAN, B.Com, F.C.A.,** and our correspondent **Smt. N. MANGAINATARAJAN, M.Sc.,** for giving motivation and providing all necessary facility for the successful completion of my project.

We wish to express my sincere thanks to our respected Director Administration, **Dr. K.K. RAMASAMY, M.E., Ph.D.,** for all the blessing and help provided during required time to complete the same.

We would like to thank out respected Principal **Dr. M. PREM KUMAR, M.E., Ph.D.,** for their moral support and deeds in bringing out this project successfully.

We extend our gratefulness to **Mr. M. SUNDARAM, M.E.,** Head of the Department**,** for the able guidance and useful suggestions, which helped us for completing project work in time.

We express our sincere thanks to **Mr. M. SUNDARAM, M.E.,** project Coordinator for the useful suggestions, which helped us for completing the project in time.

We would like to extend my sincere thanks to **Mr. M. SUNDARAM, M.E.,** Supervisor, for giving me an opportunity to do this project and also for his inspiring guidance, generous help and support.

We would like to extend my sincere thanks to **all our department staff members, and our parents** for their encouragement and advice to do the project work with full interest and enthusiasm.

ABSTRACT

Communication is a fundamental human need. However, for individuals who are hearing and speech impaired, communicating with the rest of society presents a major challenge. Sign Language serves as a visual form of communication for the deaf and mute, yet it remains unfamiliar to a significant portion of the general population. This communication barrier often leads to isolation and a lack of inclusivity for these individuals in social and professional settings. To address this issue, the project titled **"Sign Language Interpreter for Hearing Impaired"** offers a software-based solution that translates spoken English into Indian Sign Language (ISL) using images and GIFs. The system uses live speech input captured via a microphone and processes it using speech recognition libraries. If the recognized speech matches predefined phrases in the ISL dictionary, corresponding ISL animations (GIFs) are displayed. For unrecognized phrases, the system breaks down the sentence into individual characters and displays corresponding ISL alphabet signs using static images. The application is developed using Python 3.7.4 and makes use of various libraries such as Speech Recognition, NumPy, OpenCV, PIL, Matplotlib, and GUI frameworks like Tkinter and EasyGUI. It supports both online (Google API) and offline (CMU Sphinx) recognition methods, ensuring flexibility and usability in both connected and disconnected environments. The system's design emphasizes user-friendliness, low cost, and scalability, aiming to provide an accessible communication bridge for the hearing impaired. This project contributes toward an inclusive society by enhancing communication for differently-abled individuals. Future improvements can include gesture animation generation, regional language support (e.g. , Tamil or Hindi), and mobile app deployment for broader reach and real-time interaction.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATION

Acronym/Abbreviation	Full Form
AI	Artificial Intelligence
API	Application Programming Interface
ASR	Automatic Speech Recognition
CMU	Carnegie Mellon University
DFD	Data Flow Diagram
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HCI	Human-Computer Interaction
IDE	Integrated Development Environment
ISL	Indian Sign Language
JPEG	Joint Photographic Experts Group
NLP	Natural Language Processing
PIL	Python Imaging Library
SR	Speech Recognition
Tkinter	Tool Kit Interface (Python GUI Library)

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Sign Language Interpreter for Hearing Impaired is a Python-based software solution designed to address the communication challenges faced by the hearing and speech impaired in India. While Indian Sign Language (ISL) serves as the primary mode of communication within the hearing-impaired community, its limited awareness among the general public creates significant social, academic, and professional barriers. This project aims to bridge that gap by converting live voice input into ISL using speech recognition and visual translation.

The system captures spoken input through a microphone, converts it into text via both online (Google Speech Recognition API) and offline (CMU Sphinx) methods, and then displays corresponding ISL gestures through images or GIF animations. It prioritizes complete phrase matching from a predefined ISL dataset for accurate translation, but also falls back to individual character representation if a phrase match is not found, ensuring broader coverage of spoken content. Designed using Tkinter and EasyGUI, the application features a simple and intuitive graphical interface that does not require advanced hardware or internet access, making it cost-effective and widely accessible. Its dual-mode functionality allows it to work in both connected and disconnected environments, offering flexibility and practicality for real-world use.

This system can be effectively implemented in schools, public service areas, or households to improve communication and promote inclusivity for the hearing-impaired population. Furthermore, the project lays the foundation for future development such as integrating regional languages like Tamil and Hindi, incorporating deep learning models for dynamic gesture generation, and deploying the solution on mobile platforms for increased portability.

1.2 OBJECTIVES

The primary objective of this project is to bridge the communication gap between the hearing-impaired community and the general public by developing a real-time translation system that converts spoken English into Indian Sign Language (ISL). This system aims to provide a simple, cost-effective, and accessible solution using Python-based programming and open-source libraries. It supports both phrase-based and letter-based ISL translation, when a complete phrase is recognized, it displays the corresponding animated ISL gesture using GIFs. In cases where phrases do not directly match the predefined ISL dataset, the system breaks them into individual characters and displays each one using ISL alphabet images.

One of the unique aspects of the system is its dual-mode speech recognition functionality—leveraging Google's Speech Recognition API for online environments and CMU Sphinx for offline use—ensuring its usability in various scenarios regardless of internet availability.

The project also focuses on providing an intuitive and user-friendly graphical user interface through Tkinter and EasyGUI, making it suitable for use in public institutions, educational centers, and homes. By utilizing only freely available tools and resources, the system remains highly affordable and scalable, allowing future enhancements such as support for regional languages like Tamil and Hindi, integration of real-time 3D gestures, or even mobile platform deployment. Ultimately, the objective is to promote inclusivity, enable independent communication for the hearing impaired, and contribute to building a more accessible and understanding society

CHAPTER 2

LITERATURE SURVEY

1. TITLE	**:**	KUNet - An Optimized AI-Based Bengali Sign Language Translator for Hearing Impaired and Non-Verbal People [1]
AUTHOR(S)	**:**	Abdullah Al Jaid Jim, Ibrahim Rafi, Jun Jiat Tiang, Uzzal Biswas, Abdullah-Al Nahid
YOP	**:**	2024
JOURNAL	**:**	IEEE Access, October 2024

LITERATURE SUMMARY:

This paper introduces KUNet, an optimized AI-based architecture designed to recognize and translate Bengali Sign Language (BdSL) for hearing-impaired and non-verbal individuals. The authors highlight the inherent complexity in BdSL due to its large vocabulary and diverse gesture expressions, making it one of the most difficult sign languages to process. The model utilizes deep learning and optimized convolutional architectures to accurately classify static and dynamic hand gestures. One of the significant contributions of this research is addressing the cost and accessibility barriers faced by users relying on human interpreters.

The authors trained the model on a dedicated dataset containing thousands of labeled sign images and achieved superior accuracy compared to other benchmark models. While the system focuses on BdSL, its core architecture and training principles can be extended to other sign languages, including ISL. This work is highly relevant to our project, as it supports the idea of using software-based recognition systems to offer communication support without human intervention. However, our system relies on a dictionary and predefined images/GIFs rather than on deep learning, making it more accessible and hardware-independent.

2. TITLE	**:** MediSign – An Attention-Based CNN-BiLSTM Approach for Classifying Word-Level Signs for Patient-Doctor [3]
AUTHOR(S)	**:** Md. Amimul Ihsan, Abrar Faiaz Eram, Lutfun Nahar, Muhammad Abdul Kadir
YOP	**:** 2024
JOURNAL	**:** IEEE

LITERATURE SUMMARY:

This paper presents MediSign, an intelligent medical sign language interpreter tailored to support patient-doctor communication for hearing-impaired individuals. The study identifies that one of the most crucial areas requiring reliable translation is healthcare, where understanding symptoms and prescriptions is vital. MediSign introduces an attention-based CNN-BiLSTM model trained on a dataset of 30 word-level medical signs frequently used in clinical interactions. The CNN layers extract spatial features from gesture images, while the BiLSTM captures temporal dependencies. The attention mechanism enhances focus on the most relevant parts of the input, boosting classification accuracy. MediSign achieved impressive performance in both training and real-world hospital settings. It also demonstrated robust classification across different lighting and background conditions.

While this paper uses a machine learning approach with sequential data, our system uses a rule-based structure where phrases are matched to pre-recorded ISL GIFs or split into alphabets. Nevertheless, MediSign offers inspiration for future upgrades where healthcare-specific ISL phrases can be incorporated to enhance utility in medical environments. Its focus on practical application and its integration into the health sector give us insights on how our translator can be extended beyond generic usage.

3. TITLE **:** Technical Approaches to Chinese Sign Language Processing[9]
AUTHOR(S) : Suhail Muhammad Kamal, Yidong Chen, Shaozi Li
YOP **:** 2024
JOURNAL **:** IEEE

LITERATURE SUMMARY:

This comprehensive review focuses on the technical landscape of Chinese Sign Language (CSL) processing, presenting a historical and technical roadmap of how sign language systems have evolved over time. The paper explores major categories of sign recognition including static image processing, dynamic gesture recognition, and neural network-based systems. A strong emphasis is placed on the importance of datasets, with the authors citing the lack of large-scale, annotated CSL databases as a bottleneck for progress.

Techniques discussed include CNNs, RNNs, 3D ConvNets, and transformers — each offering strengths in feature extraction and time-series classification. Real-time gesture tracking and non-manual cue integration (facial expressions, head pose) are also explored. While our system does not utilize these advanced AI models, the paper emphasizes the importance of combining gesture with grammar, a principle that aligns with our text parsing and dictionary matching approach. The paper also outlines the necessity for systems to be low-cost, fast, and adaptable — goals we strive for in our ISL translator.

This review serves as a comparative benchmark to position our system among international efforts and guides future enhancements through learned best practices.

.

4. **TITLE** **:** Speech to Indian Sign Language (ISL) Translation System[5]
AUTHOR(S) **:** Pankaj Sonawane, Parth Patel, Karan Shah, Shikhar Shah,Jay
YOP **:** 2021
JOURNAL **:** International Conference on Computing

LITERATURE SUMMARY:

This paper forms the base of our final-year project. It proposes a framework that accepts voice input and translates it into corresponding Indian Sign Language gestures. The system combines speech recognition (via Google API), natural language processing (NLP), and a gesture rendering mechanism using motion-captured data. It utilizes rule-based parsing to map spoken sentences to ISL-compliant grammar and then animates gestures using a 3D avatar developed in Unity. For unmatched words, fingerspelling is used, converting each letter into hand signs.

Although their implementation relies on Kinect hardware and Unity 3D animations, the concept closely mirrors our architecture. Our implementation uses Python with Tkinter and EasyGUI to display pre-recorded GIFs and character images. This simplification enables offline operation and easy deployment without hardware dependencies.

The methodology described in this paper, including stopword removal, stemming, and dictionary-based translation, has been adapted into our project, providing a lightweight and user-friendly alternative to the original 3D-based framework. This study validates the technical feasibility of such a system and supports its relevance for real-world use.

5. **TITLE** : Real-Time ASL Recognition Using CNN [7]
AUTHOR(S) : Commonly cited academic work on ASL recognition
YOP : 2023
JOURNAL : IEEE

LITERATURE SUMMARY:

This paper presents a real-time American Sign Language (ASL) gesture recognition system using convolutional neural networks (CNNs) and transfer learning. The authors emphasize the advantage of using pretrained models such as MobileNetV2 to improve accuracy and reduce training time, especially when datasets are limited.

The system processes video input to identify ASL gestures and classify them into corresponding text outputs in real time. It focuses on optimizing inference time for deployment on mobile and low-power devices. The use of lightweight CNNs and edge computing makes it suitable for real-time applications like mobile apps for the hearing impaired.

Although this research focuses on ASL, the model and methodology can be extended to other sign languages, including ISL. In our project, we take a non-AI approach by matching recognized speech text to a predefined ISL dictionary. While we do not use CNNs or video input, the paper demonstrates how deep learning can further enhance sign language applications — potentially informing future versions of our system that may explore real-time webcam-based ISL recognition.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the current landscape, communication between hearing-impaired individuals and the general public is significantly hindered due to the lack of a common medium. Sign Language, though rich and expressive, is not widely known by the general population. In India, Indian Sign Language (ISL) serves as the primary mode of communication for hearing and speech-impaired individuals. However, there is a scarcity of systems that can automatically translate spoken language into ISL.

Most of these systems either use gesture recognition from video input (camera-based hand tracking) or involve the use of wearable sensor-based gloves. converting speech or audio input into visual signs for better accessibility. Moreover, many of the existing solutions are hardware-intensive, relying on depth sensors, motion capture tools like Microsoft Kinect, or other costly equipment. Such systems are not scalable or affordable for daily use by the general population, particularly in resource-limited settings like rural areas.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- **Limited Language Support:** These systems are often focused on American Sign Language (ASL), and ISL support is very minimal.
- **Unidirectional Communication:** Many systems focus on converting gestures to text or speech, but not from audio to gestures.
- **Lack of Offline Capability:** Most solutions require an internet connection for processing, which limits accessibility in remote regions.
- **High Development Cost:** Real-time 3D modeling or machine learning-based gesture generation requires significant computational resources and technical expertise.

3.2 PROPOSED SYSTEM

The proposed system titled **“Sign Language Interpreter for Hearing Impaired”** is designed to bridge the communication gap by translating live voice inputs into corresponding Indian Sign Language representations. The system uses Python-based speech recognition to capture user input via a microphone and then matches the spoken text with a predefined dictionary of ISL phrases. If a match is found, a GIF animation of the phrase is displayed.

If no match exists, the sentence is broken into individual letters, and each character is represented by its respective ISL alphabet image. The application uses open-source libraries like SpeechRecognition, EasyGUI, Tkinter, OpenCV, and PIL to offer a graphical interface and visual output.

This system does not require any special hardware a basic microphone and display screen are sufficient. It can function in both online mode (Google Speech Recognition API) and offline mode (CMU Sphinx), ensuring availability even in low-connectivity areas. This makes the system cost-effective, easy to use, and scalable for daily communication between the speech-impaired and the general population.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- **No Special Hardware Required:** Runs on any standard desktop/laptop with a microphone and display.
- **ISL Focused:** Unlike most systems which focus on ASL, this project uses Indian Sign Language, catering specifically to the Indian population.
- **Dual Mode Support:** Works online using Google Speech API and offline using CMU Sphinx.
- **Easy to Use GUI:** Built with Tkinter and EasyGUI, allowing intuitive navigation and usage.
- **Cost-Effective:** No expensive software or tools are needed; it can be deployed at minimal cost.

3.3 FEASIBILITY STUDY

Feasibility studies are critical in determining whether a proposed system is technically, operationally, and financially viable. It ensures the success of the system by identifying potential risks, benefits, and costs early in the development process.

3.3.1 TECHNICAL FEASIBILITY

The proposed system is highly feasible from a technical standpoint. It is built using Python 3.7.4 and several widely-used open-source libraries, including:

- **SpeechRecognition:** For voice-to-text conversion.
- **CMU Sphinx:** For offline voice recognition.
- **Google API:** For high-accuracy online speech processing.
- **Tkinter and EasyGUI:** For creating a graphical user interface.
- **PIL and OpenCV:** For displaying static and animated ISL visuals.
- **Matplotlib and NumPy:** For image rendering and matrix-based processing.

The only technical requirement is a working microphone and a computer system with basic specifications. No GPU or high-performance computing environment is necessary. The system is lightweight and can be deployed on Windows or Linux platforms without significant modification. Since the modules are independent, new features or languages can be integrated with ease.

3.3.2 OPERATIONAL FEASIBILITY

This system has been designed with user simplicity and inclusivity in mind. It is intended for use by non-technical users, NGOs, hospitals, special schools, and everyday individuals interacting with hearing-impaired persons.

The system flow is straightforward:

- Launch the application
- Click “Start Listening”
- Speak a sentence
- Watch ISL translation via GIF or alphabet images

There is no complex setup or user training required. Its GUI ensures that even a first-time user can operate it easily. Since it runs on basic hardware and requires no external database or server, it is ideal for use in offline, remote, and low-budget environments as well.

3.3.3 FINANCIAL FEASIBILITY

From a financial perspective, the proposed system is extremely economical. All the software tools and libraries used are open-source and freely available. There is no need for commercial licenses or subscriptions. The development cost involves only the time and effort of the developers. Once built, the system can be deployed without any additional infrastructure. Unlike sensor-based or 3D-rendered systems which require costly hardware or licenses, this project works on existing devices.

Potential deployment costs include:

- Basic computer/laptop with a microphone
- Pre-installed Python environment
- Storage space for images and GIFs (minimal)

Thus, the solution is highly feasible and sustainable, even for non-profit organizations, schools, and healthcare centers.

CHAPTER 4

PROJECT DESCRIPTION

4.1 PROBLEM DEFINITION

Communication is a fundamental aspect of daily life, and yet, for the hearing and speech-impaired population, communicating effectively with others poses a serious challenge. Sign Language, especially Indian Sign Language (ISL), is their primary mode of communication. However, the majority of the general population is unfamiliar with ISL, creating a significant communication barrier.

This leads to a situation where individuals with hearing impairments find it difficult to interact in schools, hospitals, workplaces, and other social settings.The lack of interpreters and the absence of accessible and cost-effective solutions further exacerbate the problem. In most public places, interpreters are not available, and technological alternatives are either too expensive or rely heavily on internet connectivity and high-end hardware.

These limitations make it crucial to develop a scalable and easy-to-use system that can bridge this communication gap. Thus, this project aims to develop a software-based solution titled **“Sign Language Interpreter for Hearing Impaired”** which can interpret spoken English sentences into ISL using visual representations in the form of images and GIFs. The system seeks to promote inclusion and accessibility for the differently-abled community in everyday communication.

4.2 OVERVIEW OF THE PROJECT

The proposed system is a speech-to-sign-language translator that captures live speech using a microphone and converts it into Indian Sign Language visuals. It leverages Python’s speech recognition libraries to detect the spoken input. Based on whether the recognized sentence exists in the predefined ISL dictionary, it either plays a corresponding animated GIF or breaks the sentence into individual characters and shows static alphabet signs.

This dual-layer translation approach ensures that even if a specific phrase is not found in the dictionary, the system will still communicate the message by spelling it out letter-by-letter in sign language.

4.2.1 CORE COMPONENTS

The key components of the project are categorized as follows:

1. **Speech Recognition Module:**
   - Captures user input through a microphone.
   - Uses speech_recognition library with either Google Speech API (online) or CMU Sphinx (offline).
   - Converts spoken audio into text for further processing.
2. **Text Matching and NLP:**
   - The recognized text is cleaned and formatted (lowercase, punctuation removed).
   - The system checks if the entire phrase matches with a predefined ISL dictionary.
3. **Sign Language Display Engine:**
   - If a match is found, plays the respective ISL GIF animation.
   - If no match is found, it processes the input as individual characters and displays their corresponding ISL alphabet images.
   - Uses PIL, OpenCV, Matplotlib, and Tkinter for rendering images and animations.
4. **Graphical User Interface (GUI):**
   - Built using Tkinter and EasyGUI.
   - Allows users to click and interact with the system easily.
   - Supports user-friendly buttons like “Start Listening” and “Exit”.

5. **Dual Mode Recognition:**

    - Internet-connected systems use high-accuracy Google API.
    - Offline systems use Sphinx, making the system adaptable to all environments.

4.2.2 EXISTING CHALLENGES ADDRESSED

This project directly addresses multiple limitations found in the existing systems:

- **Hardware Independence:** No need for Kinect, Leap Motion, or other expensive sensors.
- **Internet Optional:** Works even in offline mode using Sphinx, unlike most solutions that rely heavily on cloud-based APIs.
- **ISL-Specific:** Unlike many international solutions focused on ASL, this system is based entirely on Indian Sign Language.
- **Cost-Effective:** Built using only open-source software and runs on any regular PC or laptop.
- **User-Centric Design:** Simple GUI ensures usability by anyone without technical knowledge.
- **Dictionary Expansion:** Users can add more phrases or alphabets, making the system extensible and adaptive.
- **Educational Use:** Can be used as a teaching tool in schools for special children to learn alphabets and basic communication.

These improvements aim to bring real, practical value to communities with limited access to high-end technology, interpreters, or modern infrastructure.

4.2.3 MODULES AND ARCHITECTURE

The system consists of several well-defined modules that function independently and collectively to achieve the project’s goal:

- **Speech Input Module:** Records live voice from the microphone.
- **Recognition Module:** Converts audio to text using speech-to-text libraries.
- **Dictionary Matcher:** Compares recognized text with the ISL phrase.
- **Image Display Module:** Displays either a GIF for phrases or image .
- **GUI Module:** Handles user interaction.

System Architecture Overview:

1. **Input Stage:** User speaks a sentence through the microphone.
2. **Processing Stage:**Speech-to-text conversion → Text cleaning → Phrase
3. **Output Stage:** If a GIF is available, display it. Else, convert text to alphabet signs and show letter images in sequence.

This modular architecture ensures flexibility, ease of debugging, and scope for future upgrades.

4.2.4 SOFTWARE AND TOOLS USED

The entire system is developed using the **Python 3.7.4** programming language. Python was chosen due to its simplicity, vast library support, and ease of prototyping. The following libraries and tools are used

- **SpeechRecognition:** For capturing and converting speech to text.
- **CMU Sphinx:** Offline voice recognition.
- **Google API:** Online speech recognition.
- **Tkinter & EasyGUI:** GUI design and interaction.
- **OpenCV and PIL:** Image processing and animation rendering.

- **NumPy and Matplotlib:** Array and image display functionality.
- **OS Module:** For file handling and image/GIF retrieval.

These tools are all open-source, making the project financially viable for wide-scale deployment. No paid licenses or platforms are required to develop, test, or use the application.

4.2.5 FUTURE ENHANCEMENT

The current system is designed to handle English voice input and translate it into Indian Sign Language using pre-defined phrase GIFs or character images. However, there are several directions in which the system can be enhanced further:

- **Regional Language Support:** Integration of Tamil, Hindi, and other regional languages as input.
- **Reverse Translation:** Allow sign gestures to be captured via webcam and translated into spoken output (ISL to English).
- **Real-Time Gesture Rendering:** Replacing static images with AI-based 3D rendering of ISL signs.

CHAPTER 5
MODULES DESCRIPTION

The proposed system is designed in a modular approach, where each module is responsible for a specific function in the process of translating speech into Indian Sign Language (ISL). This modular structure ensures better maintainability, scalability, and easier debugging or enhancements in the future. The system is divided into the following four main modules:

5.1 AUDIO INPUT AND PREPROCESSING MODULE

This is the first and most crucial component of the system. The primary role of this module is to capture live speech through the microphone and convert it into textual data for further processing.

- The module uses the speech_recognition library to interface with the microphone.
  Two different engines are used for conversion:

- **Google Speech Recognition API (online mode):** Offers high accuracy and supports various accents and sentence structures.
- **CMU Sphinx (offline mode):** Enables voice recognition in the absence of internet connectivity.
- The module also calibrates for background noise to ensure better voice detection.
- The audio is recorded and then transformed into lowercase text, stripping unnecessary punctuation and noise.

This module ensures the raw input is cleaned and ready for meaningful processing. It acts as the foundation for the subsequent modules by reliably converting speech into text.

5.2 TEXT PROCESSING AND MAPPING MODULE

Once the spoken input has been converted into text, the Text Processing and Mapping module takes over. Its main responsibilities include:

- **Text Cleaning:** Removing punctuation and converting all characters to lowercase.
- **Phrase Matching:** The cleaned text is compared against a pre-defined dictionary of commonly used ISL phrases.
- If a match is found, the system fetches the corresponding animated GIF from the ISL library.
- If no direct match is found, the sentence is broken down into individual characters.
- **Fallback Handling:** When full-phrase matches fail, each character is mapped to its corresponding ISL alphabet image stored in the letters directory.

This mapping process ensures the message is never lost. Whether the sentence exists in the database or not, the system will still visually represent it in a letter-by-letter format using sign language.

5.3 SIGN VISUALIZATION MODULE

- **GIF Display:** For phrases that exist in the ISL dictionary, an animated GIF is played using Tkinter and PIL (Python Imaging Library). The animation runs in a pop-up window and closes automatically after a few seconds.
- **Static Image Display:** If only characters are available, the system loads each character's corresponding sign image from the letters folder and displays them sequentially using Matplotlib or Tkinter.

5.4 CONTROLLER AND GUI MODULE

This is the central control module that integrates all the other components and presents them to the user in a friendly interface.

- Built using **Tkinter** and **EasyGUI**, it handles:
- Start screen with a welcome message.
- Buttons for “Start Listening” and “Exit.”
- Automated window control for sign display.
- The GUI is designed for ease of use, especially for users who may not be technically skilled.
- The controller ensures the system flow is maintained:

  1. Capture audio
  2. Convert to text
  3. Match phrase or character
  4. Display corresponding ISL visuals

This module provides an interactive and accessible bridge between the user and the system's core functionalities.

CHAPTER 6
REQUIREMENTS ENGINEERING

6.1 SOFTWARE REQUIREMENTS

To develop and execute the “Sign Language Interpreter for Hearing Impaired” system, the following software requirements are necessary:

- **Operating System:** Windows 10 or higher (Compatible with Linux as well)
- **Programming Language:** Python 3.7.4
- **IDE/Text Editor:** VS Code / PyCharm / Sublime Text
- **Libraries and Packages:**

  1. speech_recognition – For audio-to-text conversion
  2. pyaudio – For capturing microphone input
  3. tkinter – For building GUI elements
  4. PIL (Pillow) – For image and GIF handling
  5. matplotlib – For displaying letter signs
  6. numpy – For image processing
  7. easygui – For simple user interaction windows
  8. os, sys, string, and cv2 (OpenCV) – For backend file and image handling

- **Speech Engines:**

  1. Google Speech Recognition API (Online)
  2. CMU Sphinx (Offline support)

- **File Support:**

  1. .jpg images for alphabet signs
  2. .gif animations for sentence-level ISL gestures

6.2 FUNCTIONAL REQUIREMENTS

The following functional requirements define the core features and operations the system must perform:

- **FR1 – Audio Capture:**
    1. The system shall capture live voice input through a microphone.
    2. It must allow ambient noise adjustment for better recognition.
- **FR2 – Speech Recognition:**
    1. The system shall convert audio into text using either online or offline speech recognition engines.
- **FR3 – Phrase Matching:**
    1. The system shall match the recognized text with a predefined ISL dictionary of phrases.
- **FR4 – Fallback to Character Display:**
    1. If a full phrase is not available, the system shall convert the text to character-level ISL alphabet images.
- **FR5 – GIF and Image Display:**
    1. The system shall display appropriate animated GIFs for known phrases.
    2. It must sequentially display alphabet signs for unknown sentences.
- **FR6 – GUI Interaction:**
    1. The system shall provide buttons for “Start Listening” and “Exit.”
    2. It shall visually display the output in a user-friendly window.

CHAPTER 7
SOFTWARE DESCRIPTION

7.1 PROGRAMMING LANGUAGE: PYTHON 3.7.4

The core development of the system has been done using **Python 3.7.4**. Python is a powerful high-level programming language that is widely used in artificial intelligence, machine learning, and image processing applications due to its readability and extensive library support. It is an interpreted language that provides a dynamic type system and automatic memory management.

Python was chosen for this project because:

- It has large community support.
- It is easy to integrate with multimedia libraries and speech APIs.
- It has a vast ecosystem of third-party modules.
- It supports rapid development with fewer lines of code.

Python's standard syntax makes it ideal for developing educational, real-time, and AI-integrated applications like speech-to-sign translation systems.

7.2 LIBRARIES AND TOOLS USED

To ensure that the system meets all its functional requirements, various libraries and tools are incorporated. The following are the key libraries and their specific roles in this project.

7.2.1 SPEECHRECOGNITION

The speech_recognition library is the most crucial component in this project for converting audio input into text. It supports multiple speech recognition engines and APIs including:

- Google Web Speech API (Online)
- CMU Sphinx (Offline)

Key Features:

- Easy integration with Python microphone support.
- High accuracy in recognizing human speech.
- Compatibility with both short and long speech inputs.
- Error handling for network and unknown audio cases.

In this project, it enables dual-mode recognition – switching between online and offline recognition depending on internet availability.

7.2.2 NUMPY

NumPy (Numerical Python) is used extensively for handling and manipulating image arrays. It provides a high-performance multidimensional array object and tools for working with these arrays.

Roles in the Project:

- Reading images as arrays for processing.
- Performing matrix operations for image rendering and control.
- Seamlessly integrates with OpenCV and Matplotlib for backend image tasks.

Although NumPy's role is more in the background, it is essential for operations like pixel manipulation and image rendering optimizations.

7.2.3 MATPLOTLIB

Matplotlib is a data visualization library in Python but is also used to render static ISL letter images in this project. It supports embedding plots and images in GUIs built with Tkinter.

Roles in the Project:

- Displaying each letter image in a popup format when phrases are not found.
- Rendering alphabet visuals for unknown text inputs.
- Providing pause and transition effects between alphabet displays.

The plt.pause() function from Matplotlib allows for timed image visualization, which plays an important role in showing each alphabet sign sequentially.

7.2.4 OPENCV (cv2)

OpenCV (Open Source Computer Vision Library) is one of the most popular libraries for real-time computer vision. In this project, it plays a minor role but is essential for certain display and image processing functionalities.

Roles in the Project:

- Processing GIF frames and static images for display.
- Supporting conversion and manipulation of image arrays.
- Enhancing compatibility across image formats (.jpg, .png).

Even though OpenCV is more often used for gesture recognition and camera input, in this case, it helps ensure flexible rendering of sign visuals during output display.

7.2.5 TKINTER AND EASYGUI

Both Tkinter and EasyGUI are Python libraries used to build graphical user interfaces. They offer tools to build simple, interactive windows and control elements like buttons and labels.

Tkinter:

- Used for building the primary GUI window.
- Enables buttons like “Start Listening” and “Exit”.
- Helps display animated GIFs or static ISL signs in a popup.

EasyGUI:

- Simplifies user input and message display.
- Provides a user-friendly experience without complex GUI coding.
- Used for the initial “Live Voice” or “Exit” menu.

Together, they help in building an accessible and beginner-friendly front end so that anyone – including users with minimal technical knowledge – can operate the system with ease.

CHAPTER 8
SYSTEM DESIGN

8.1 SYSTEM ARCHITECTURE

The system architecture of the “Sign Language Interpreter for Hearing Impaired” project is designed in a layered and modular manner to ensure efficiency, scalability, and ease of integration. The architecture involves different components working sequentially to convert voice input into Indian Sign Language (ISL) visuals.

System Layers:

1. **Input Layer:**

    - Captures speech via a microphone.
    - Uses speech_recognition to interpret voice into text using Google API or CMU Sphinx.

2. **Processing Layer:**

    - Preprocesses the recognized text by cleaning, converting to lowercase, and removing special characters.
    - Matches text with pre-defined ISL phrases.
    - If no phrase match is found, text is broken down into characters.

3. **Mapping Layer:**

    - Maps recognized phrase to animated GIFs.
    - Or maps characters to corresponding letter images stored in the “letters” folder.

4. Output Layer:

- Displays the ISL sign (GIF or image) using Tkinter or Matplotlib in a visual window.

5. Control Layer:

- The GUI built with Tkinter/EasyGUI manages user interaction.
- Includes options to Start Listening, Exit, and auto-close visualization windows.

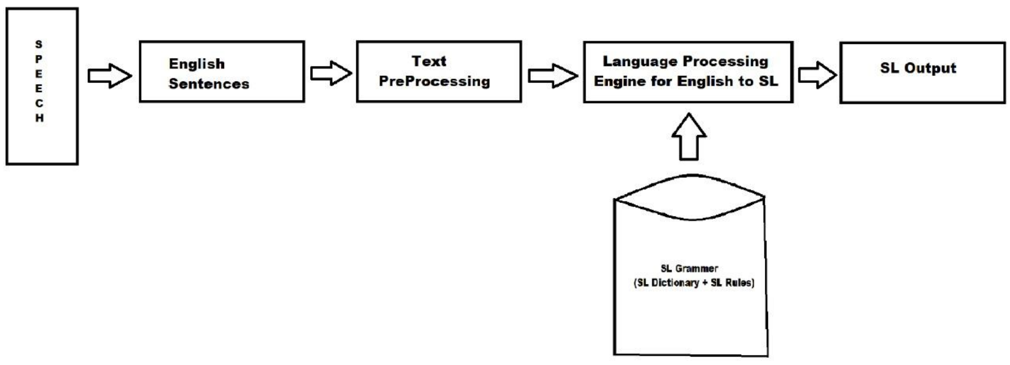

S P E E C H
English Sentences
Text PreProcessing
Language Processing Engine for English to SL
SL Output
SL Grammer (SL Dictionary + SL Rules)

FIG: 8.1 System Architecture of Sign Language Interpreter

8.2 DATA FLOW DIAGRAM (DFD)

The data flow diagram (DFD) models the system’s logical flow of data through different processing stages. It helps visualize how data is input, processed, and output in a simplified format.

8.2.1 LEVEL 0 – CONTEXT LEVEL DFD

This is the highest level in the DFD hierarchy, which shows the overall system as a single process with input and output.

Entities:

- **User (Speaker):** Provides voice input.
- **System (Process 0):** Processes input and produces ISL visual output.
- **Output Viewer:** Receives ISL signs via GIF or image.

Flow:

1. User gives voice input.
2. System recognizes and processes input.
3. Outputs sign language visuals to viewer.

8.2.2 LEVEL 1 – DETAILED DFD

The Level 1 DFD breaks down the main system into sub-processes and shows how data flows between them.

Sub-processes:

1. **Audio Capture (1.0):** Records voice from user.
2. **Speech Recognition (2.0):** Converts voice to text using APIs.
3. **Phrase Matching (3.0):** Checks text against predefined ISL phrases.
4. **Fallback to Character Mapping (4.0):** If phrase not found,maps eachch
5. **Display Output (5.0):** Shows corresponding ISL signs using images or GIFs.

Data Stores:

- **Phrase Database:** Contains phrase-to-GIF mappings.
- **Letter Image Folder:** Contains character image files.

Flow:

- Voice → Text → Phrase Match → (GIF or Character) → ISL Display

This level gives a more complete picture of the working modules and how they interact internally.

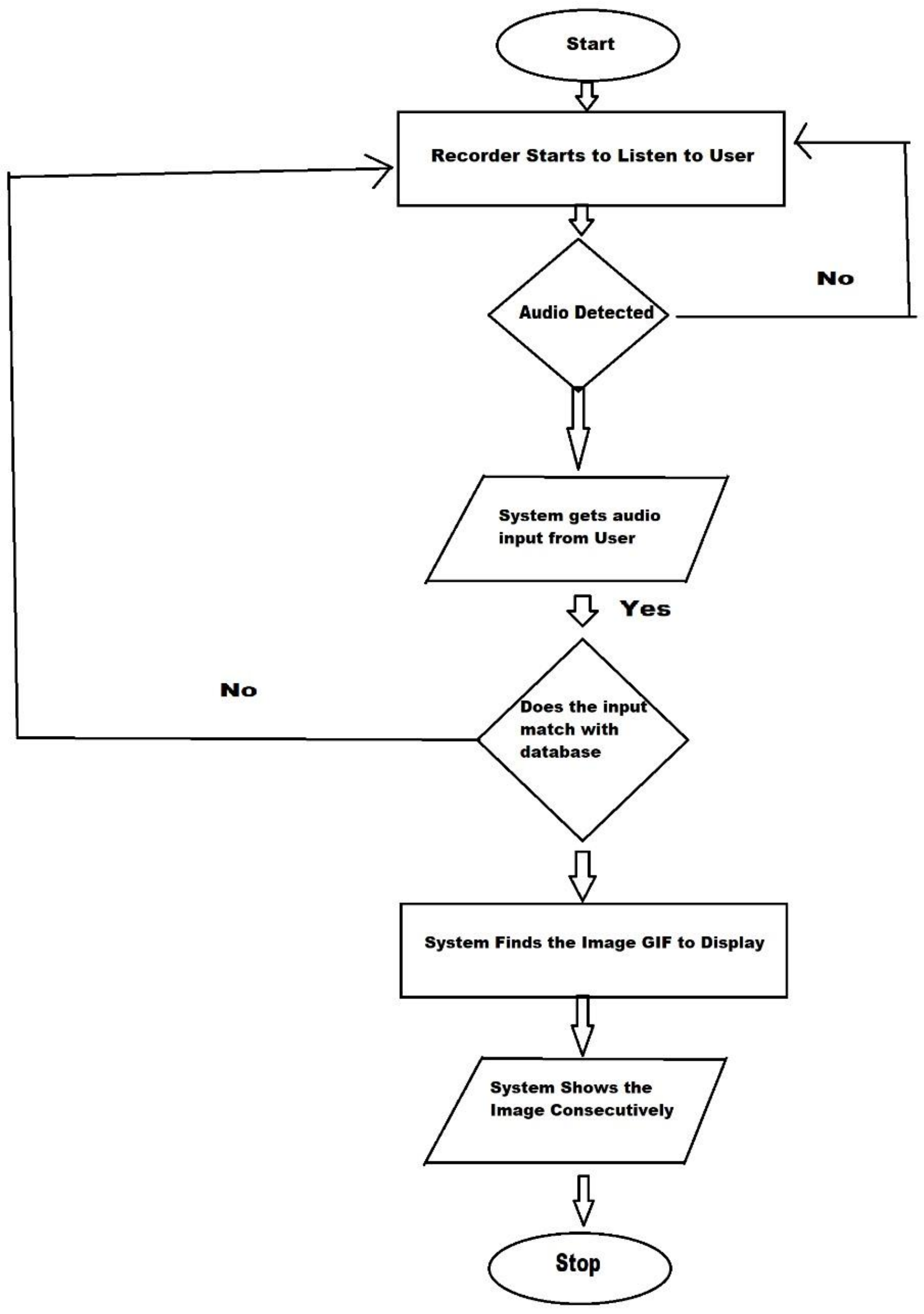

Start
Recorder Starts to Listen to User
Audio Detected
No
System gets audio input from User
Yes
No
Does the input match with database
System Finds the Image GIF to Display
System Shows the Image Consecutively
Stop

FIG: 8.2 Data Flow Diagram

CHAPTER 9
CONCLUSION AND FUTURE ENHANCEMENTS

9.1 CONCLUSION

Communication is a vital part of human interaction, yet people with hearing and speech impairments often face challenges in conveying their thoughts and understanding others. The proposed project, titled **“Sign Language Interpreter for Hearing Impaired”**, addresses this communication gap by translating spoken language into Indian Sign Language (ISL) using a software-based solution. By leveraging Python programming and widely available speech recognition technologies, this project provides a reliable and accessible tool for facilitating inclusive communication. The system captures audio from the user, processes it through speech recognition engines, and either matches the input against pre-defined ISL phrases or falls back to displaying letter-by-letter signs using alphabet images. The output is visualized through animated GIFs or static images, depending on the recognized input. A simple GUI is designed using Tkinter and EasyGUI to ensure user-friendly interaction. The major strength of this project lies in its flexibility and accessibility. It supports both online and offline modes of operation, ensuring usability in areas with limited or no internet connectivity. It is cost-effective, as it relies on free and open-source tools and does not require specialized hardware. The implementation of this project demonstrates the practical application of artificial intelligence, natural language processing, and computer vision in solving real-world social problems. Through testing and validation, the system has shown promising results in interpreting a wide range of common phrases and fallback handling for unknown inputs. It holds the potential to significantly enhance the day-to-day interaction between the hearing-impaired and the general public.

9.2 FUTURE ENHANCEMENT

While the current system performs well in its designed scope, there are several areas where future improvements and enhancements can be incorporated:

1. **Support for Regional Languages:**

The system currently works only with English input. Adding multilingual support (e.g., Tamil, Hindi, Bengali) would expand the system's usability across different parts of India.

2. **Mobile Application Development:**

A mobile version of the application would increase accessibility, especially for users who may not have access to desktop systems. Android and iOS apps can make the system portable and usable in real-time scenarios.

3. **Cloud-Based Phrase Expansion:**

A centralized cloud database can be developed to store new phrases and signs, allowing the system to learn and grow dynamically over time. Users could contribute new signs, making it a collaborative platform.

4. **3D Animation and Avatar Integration:**

Instead of using pre-recorded GIFs and static images, a virtual avatar can be introduced that dynamically performs sign gestures in 3D. This would allow for more expressive and context-aware visual output.

5. **Emotion and Context Analysis:**

With advancements in natural language understanding, the system can be trained to detect emotions, tone, or urgency in the spoken text and reflect them visually through appropriate sign intensity or facial cues.

CHAPTER 10

APPENDIX

10.1 SOURCE CODE

Main Code (main1.py)

This script handles live voice input, speech recognition, phrase matching, and display of corresponding ISL signs (GIFs or letter images). The GUI menu is built using EasyGUI.

import speech_recognition as sr, import numpy as np, import matplotlib.pyplot as plt, import cv2 from easygui, import buttonbox import os from PIL, import Image, ImageTk from itertool,s import count, import tkinter as tk ,import string

def func(): r = sr.Recognizer()

```
    isl_gif = [...]  # Same list as before — no change needed here
arr = list(string.ascii_lowercase)  # Simpler way to get a-z

# Directories
letter_images_dir = r'C:\Users\suhas\Desktop\project\app\pro\letters'
gif_images_dir = r'C:\Users\suhas\Desktop\project\app\pro\ISL_Gifs'

class ImageLabel(tk.Label):
    def load(self, im):
        if isinstance(im, str):
            im = Image.open(im)
        self.loc = 0
        self.frames = []
        try:
            for _ in count(1):
                self.frames.append(ImageTk.PhotoImage(im.copy()))
                im.seek(im.tell() + 1)
        except EOFError:
            pass

        self.delay = im.info.get('duration', 100)

        if len(self.frames) == 1:
            self.config(image=self.frames[0])
        else:
            self.next_frame()

    def unload(self):
        self.config(image=None)
        self.frames = None

    def next_frame(self):
        if self.frames:
            self.loc = (self.loc + 1) % len(self.frames)
            self.config(image=self.frames[self.loc])
            self.after(self.delay, self.next_frame)

with sr.Microphone() as source:
    r.adjust_for_ambient_noise(source)
    while True:
        print("Say something...")
        audio = r.listen(source)
```

```
try:
    a = r.recognize_sphinx(audio)
    print("You said:", a)
    a = a.lower().translate(str.maketrans('', '', string.punctuation))

    if a in ['goodbye', 'good bye', 'bye']:
        print("Goodbye!")
        break

    elif a in isl_gif:
        gif_path = os.path.join(gif_images_dir, f'{a}.gif')
        if os.path.exists(gif_path):
            root = tk.Tk()
            lbl = ImageLabel(root)
            lbl.pack()
            lbl.load(gif_path)
            root.mainloop()
        else:
            print(f"GIF not found for: {a}")

    else:
        for char in a:
            if char in arr:
                img_path = os.path.join(letter_images_dir, f'{char}.jpg')
                if os.path.exists(img_path):
                    img = Image.open(img_path)
                    img_np = np.asarray(img)
                    plt.imshow(img_np)
                    plt.axis('off')
                    plt.draw()
                    plt.pause(0.8)
                else:
                    print(f"Image not found for: {char}")
            plt.close()

        except sr.UnknownValueError:
            print("Could not understand audio.")
        except sr.RequestError as e:
            print(f"Sphinx error; {e}")
        except Exception as e:
            print(f"Unexpected error: {e}")
```

Main Code (main2.py)

```
import speech_recognition as sr

import tkinter as tk from PIL

import Image, ImageTk

import os

#import easygui as eg

isl_chars = list("abcdefghijklmnopqrstuvwxyz") # Assuming you want to
recognize lowercase letters

exit_phrases = ['thank you', 'bye', 'goodbye', 'exit']

class ImageLabel(tk.Label): def init(self, master=None): super().init(master)
self.image = None # To hold a reference to the image
```

```
def load(self, image_path):
try:
    im = Image.open(image_path)
    im.resize((800,800))
    self.image = ImageTk.PhotoImage(im)  # Keep a reference
    self.config(image=self.image)  # Set the image on the label
except FileNotFoundError:
    print(f"Image '{image_path}' not found.")
```

def recognize_speech(): r = sr.Recognizer() with sr.Microphone() as source: r.adjust_for_ambient_noise(source) print("Listening...") audio = r.listen(source) try: return r.recognize_google(audio).lower() except (sr.UnknownValueError, sr.RequestError): return None

def display_character_images(recognized_text): root = tk.Tk() lbl = ImageLabel(root) lbl.pack()

```
for char in recognized_text:
if char in isl_chars:
    img_path = os.path.join('letters', f'{char}.jpg')
    lbl.load(img_path)
    root.update()  # Update the label to show the new image
    root.after(800)  # Wait for 0.8 seconds before the next image

# Close the window after displaying all characters
root.after(800 * len(recognized_text) + 20, root.destroy)
root.mainloop()  # Start the Tkinter event loop
```

```
    def show_start_menu(): def on_start(): root.destroy() # Close the menu
start_listening()

    root = tk.Tk()
root.title("Speech to Sign Language Converter")
root.geometry("400x300")  # Set the window size
root.configure(bg="#f0f0f0")  # Background color

# Title Label
title_label = tk.Label(root, text="Hearing Impairment Assistant",
font=("Helvetica", 16, "bold"), bg="#f0f0f0")
title_label.pack(pady=20)

# if you want to add a n image
# img = ImageTk.PhotoImage(Image.open("your_image.png"))
# img_label = tk.Label(root, image=img, bg="#f0f0f0")
# img_label.pack(pady=10)

# Start Listening Button
start_button = tk.Button(root, text="Start Listening", command=on_start,
width=20, height=2, bg="#4CAF50", fg="white", font=("Helvetica", 12))
start_button.pack(pady=10)

# Exit Button
exit_button = tk.Button(root, text="Exit", command=root.quit, width=20,
height=2, bg="#f44336", fg="white", font=("Helvetica", 12))
exit_button.pack(pady=10)

root.mainloop()
```

```
def start_listening(): while True: recognized_text = recognize_speech() if
recognized_text: print(f'You said: {recognized_text}')

        # Check for exit phrases
      if any(exit_phrase in recognized_text for exit_phrase in exit_phrases):
         print("Exiting...")
         break


      display_character_images(recognized_text)
   else:
      print("No valid speech recognized. Listening continues...")


   if name == "main": show_start_menu()
```

10.2 OUTPUT SCREENSHOTS

Below are the screenshots captured during the execution of the system:

1. **Start Menu GUI** – A simple window with “Start Listening” and “Exit” options.

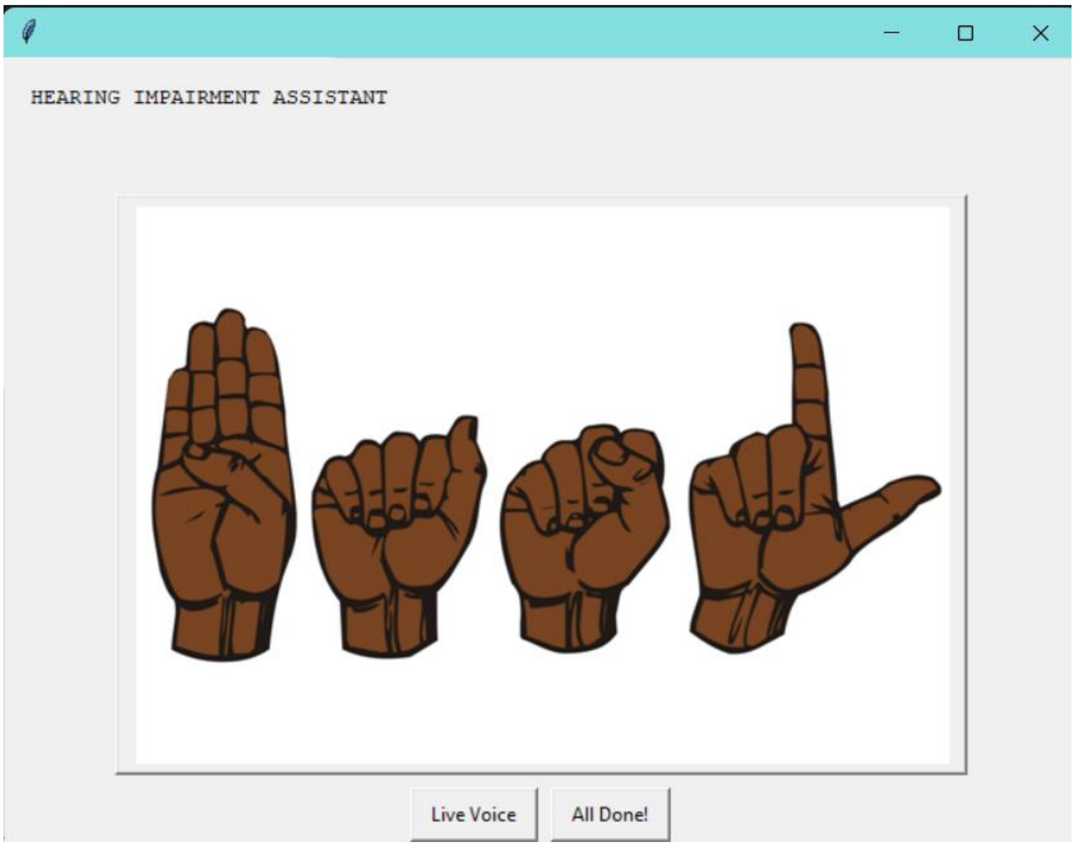
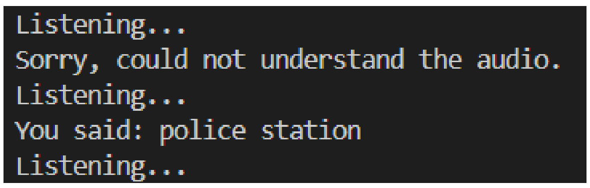


FIG:10.2.1 Starting Window

2. **Voice Recognition Console Output** – Displays recognized speech text



```
Listening...
Sorry, could not understand the audio.
Listening...
You said: police station
Listening...
```

FIG:10.2.2 Output Console

3. **Phrase Match GIF Output** – GIF displayed for recognized phrase like "goodmorning".

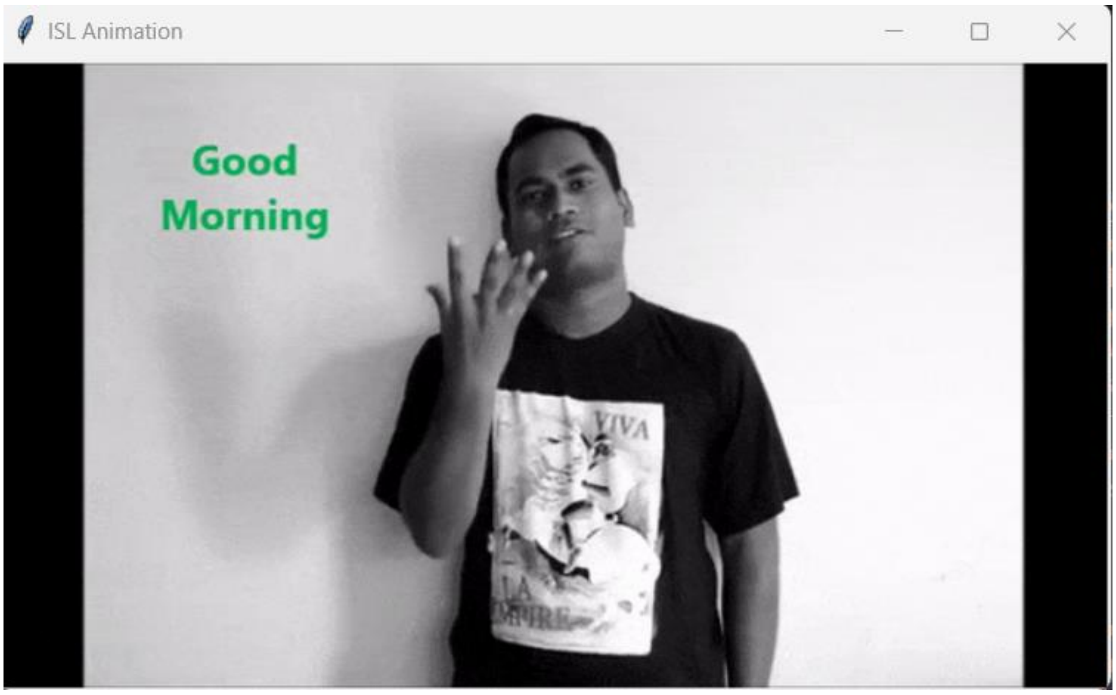


FIG:10.2.3 GIF Output

CHAPTER 11
REFERENCES

[1] Abdullah Al Jaid Jim, Ibrahim Rafi, Jun Jiat Tiang, Uzzal Biswas, Abdullah-AlNahid,*"KUNet – An Optimized AI-Based Bengali Sign Language Translator for Hearing Impaired and Non-Verbal People,"*IEEE Access, Oct 2024,DOI: 10.1109/ACCESS.2024.3474011.

[2] Bashaer A. Al Abdullah, Ghada A. Amoudi, Hanan S. Alghamdi, *"Advancements in Sign Language Recognition: A Comprehensive Review and Future Prospects,"*IEEE Access, 2024.

[3] Md. Amimul Ihsan, Abrar Faiaz Eram, Lutfun Nahar, Muhammad Abdul Kadir,*"MediSign: An Attention-Based CNN-BiLSTM Approach of Classifying Word-Level Signs for Patient-Doctor Interaction,"* Department of Biomedical Physics and Technology, University of Dhaka, 2024.

[4] OpenCVTeam,*"OpenCV Documentation,"*Available at: https://opencv.org

[5] Pankaj Sonawane, Parth Patel, Karan Shah, Shikhar Shah, Jay Shah, *"Speech To Indian Sign Language (ISL) Translation System,"* 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS),DOI: 10.1109/ICCCIS51004.2021.9397097.

[6] Python Software Foundation,*"Python 3.7.4 Documentation,"* Available at: https://docs.python.org/3.7/

[7] "*Real-Time ASL Recognition Using CNN*" by Commonly cited academic work on ASL recognition in IEEE 2023

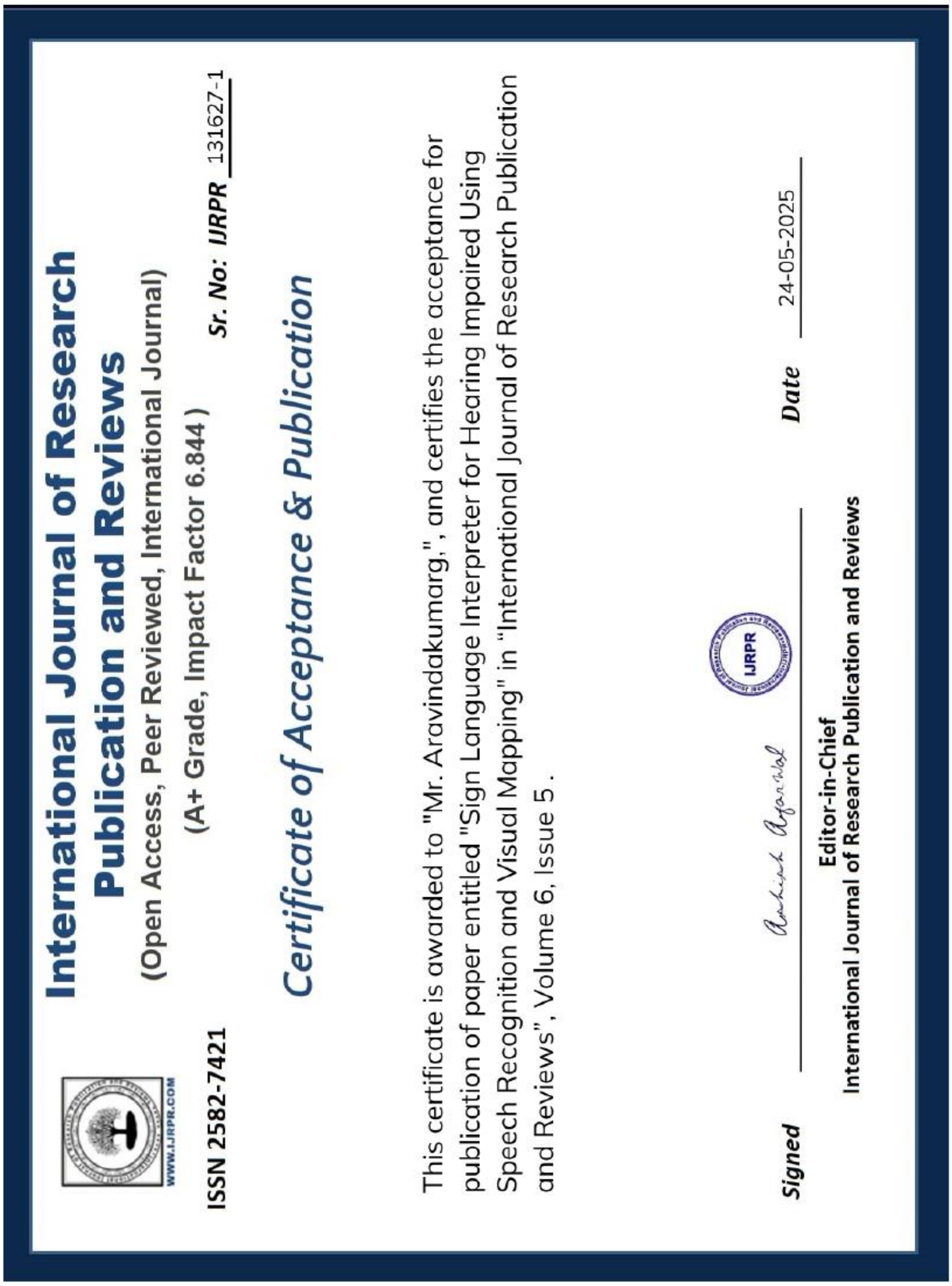
[8] R.Rumana, Reddygari Sandhya Rani, R. Prema, *“A Review Paper on Sign Language Recognition for The Deaf and Dumb,”* Department of Computer Science and Engineering, SCSVMV, Kanchipuram, 2023.

[9] Suhail Muhammad Kamal, Yidong Chen, Shaozi Li, Xiaodong Shi, Jiangbin Zheng,*“Technical Approaches to Chinese Sign Language Processing: A Review,”*Xiamen University, IEEE,DOI: 10.1109/ACCESS.2024.XXXXX (Replace with actual DOI if available).

[10] Tkinter – Python Interface to Tcl/Tk GUI Toolkit, Available at: https://docs.python.org/3/library/tkinter.html

CHAPTER 12

CERTIFICATES



WWW.IJRPR.COM

International Journal of Research Publication and Reviews

(Open Access, Peer Reviewed, International Journal)

(A+ Grade, Impact Factor 6.844)

ISSN 2582-7421 ***Sr. No: IJRPR*** 131627-1

Certificate of Acceptance & Publication

This certificate is awarded to "Mr. Aravindakumarg.", and certifies the acceptance for publication of paper entitled "Sign Language Interpreter for Hearing Impaired Using Speech Recognition and Visual Mapping" in "International Journal of Research Publication and Reviews", Volume 6, Issue 5 .

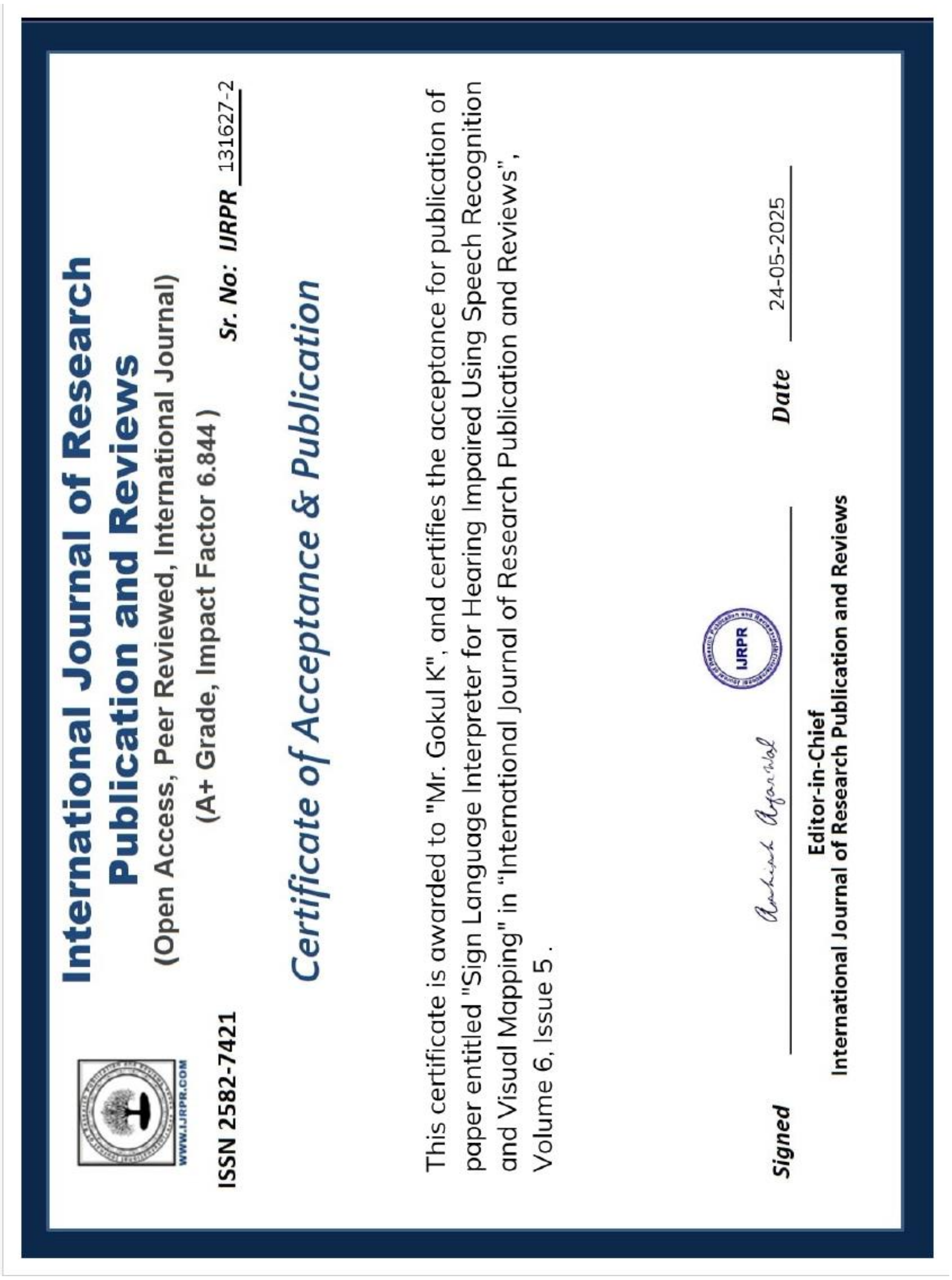
Signed [illegible]

IJRPR

Date 24-05-2025

Editor-in-Chief
International Journal of Research Publication and Reviews

FIG:12.1 Certificate of Aravindakumar G



WWW.IJRPR.COM

International Journal of Research Publication and Reviews

(Open Access, Peer Reviewed, International Journal)

(A+ Grade, Impact Factor 6.844)

ISSN 2582-7421 ***Sr. No: IJRPR*** 131627-2

Certificate of Acceptance & Publication

This certificate is awarded to "Mr. Gokul K", and certifies the acceptance for publication of paper entitled "Sign Language Interpreter for Hearing Impaired Using Speech Recognition and Visual Mapping" in "International Journal of Research Publication and Reviews", Volume 6, Issue 5 .

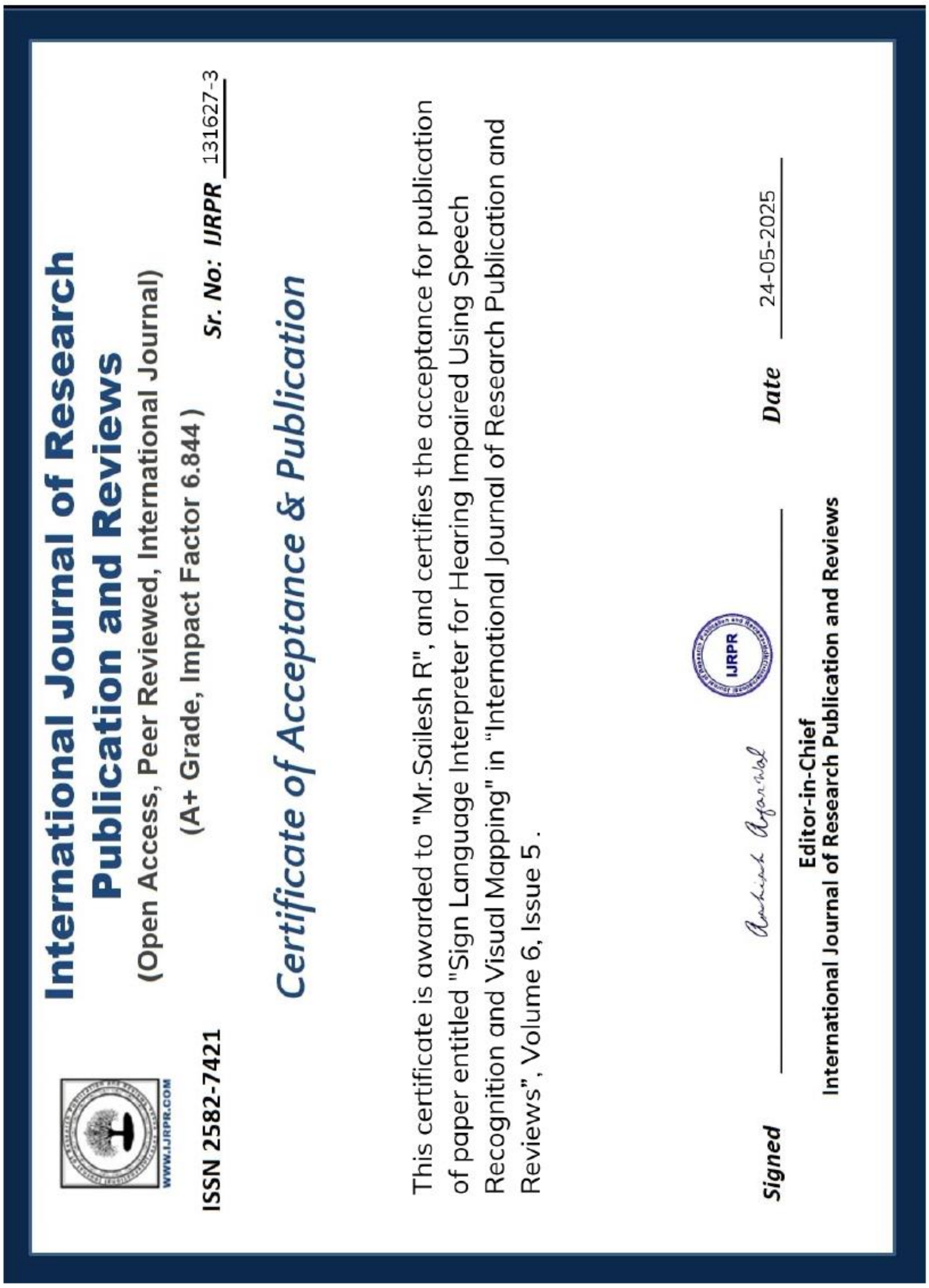
Signed [illegible]

IJRPR

Date 24-05-2025

Editor-in-Chief
International Journal of Research Publication and Reviews

FIG:12.2 Certificate of Gokul K



International Journal of Research Publication and Reviews

(Open Access, Peer Reviewed, International Journal)

(A+ Grade, Impact Factor 6.844)

WWW.IJRPR.COM

ISSN 2582-7421

Sr. No: IJRPR 131627-3

Certificate of Acceptance & Publication

This certificate is awarded to "Mr.Sailesh R", and certifies the acceptance for publication of paper entitled "Sign Language Interpreter for Hearing Impaired Using Speech Recognition and Visual Mapping" in "International Journal of Research Publication and Reviews", Volume 6, Issue 5 .

IJRPR

International Journal of Research Publication and Reviews

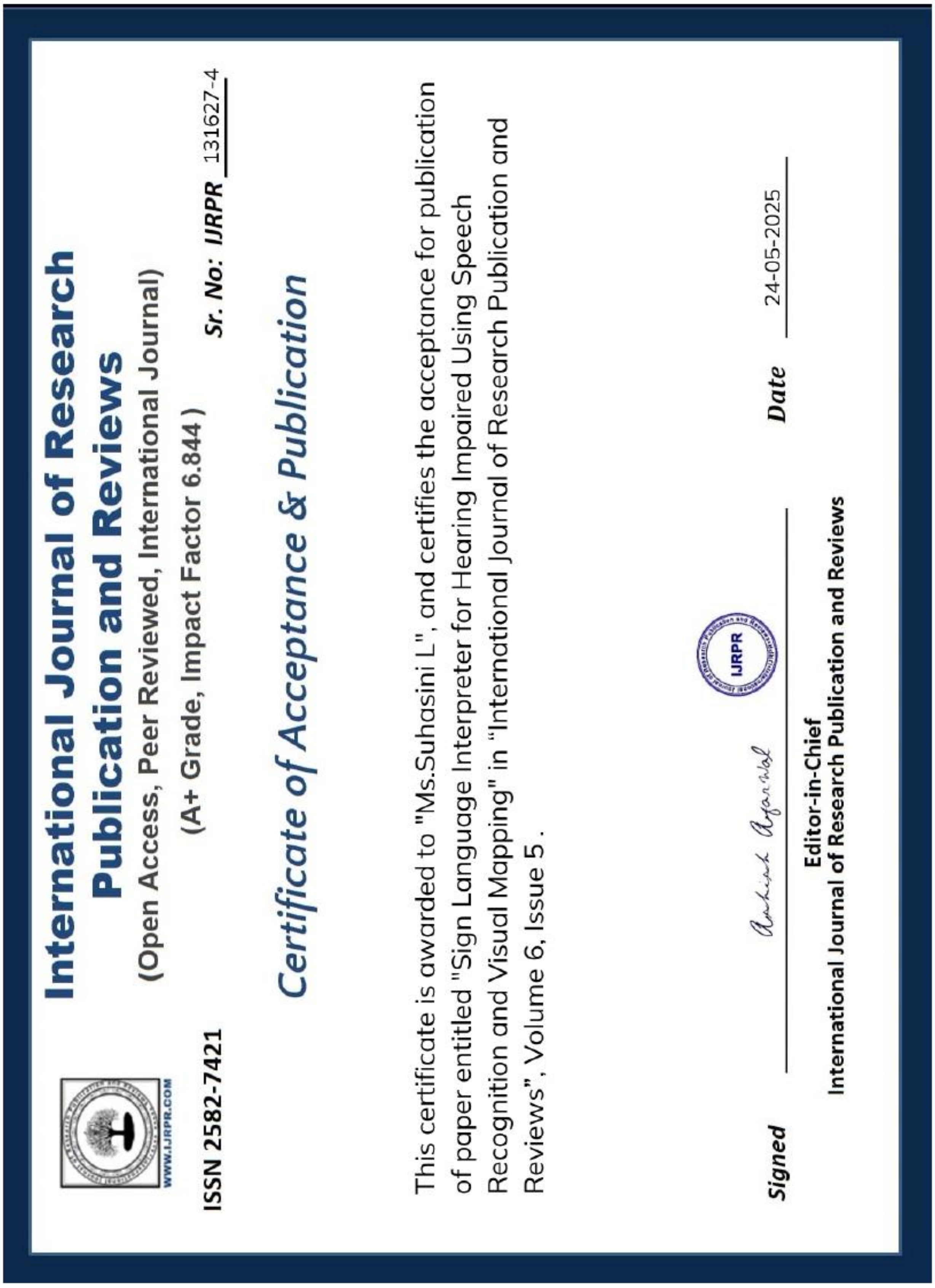
Signed [illegible]

Editor-in-Chief

International Journal of Research Publication and Reviews

Date 24-05-2025

FIG:12.3 Certificate of Sailesh R



WWW.IJRPR.COM

ISSN 2582-7421

International Journal of Research Publication and Reviews

(Open Access, Peer Reviewed, International Journal)

(A+ Grade, Impact Factor 6.844)

Sr. No:* *IJRPR 131627-4

Certificate of Acceptance & Publication

This certificate is awarded to "Ms.Suhasini L", and certifies the acceptance for publication of paper entitled "Sign Language Interpreter for Hearing Impaired Using Speech Recognition and Visual Mapping" in "International Journal of Research Publication and Reviews", Volume 6, Issue 5 .

Signed [illegible]

IJRPR

Editor-in-Chief
International Journal of Research Publication and Reviews

Date 24-05-2025

`

FIG:12.4 Certificate of Suhasini L