

# **Capturing Non-manual features of Indian Sign Language and Converting it into Text**

**A PROJECT REPORT**

*Submitted by,*

Ms. MAMATHA S	-	20211CSG0048
Ms. PRACHI	-	20211CSG0025
Ms. NARMADA RADHIKA J S	-	20211CSG0028
Ms. NITHYA T M	-	20211CSG0006

*Under the guidance of,*

**Dr. LEELAMBIKA K V**

*in partial fulfillment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**At**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

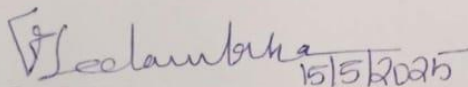
**May 2025**

## PRESIDENCY UNIVERSITY

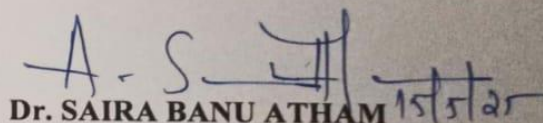
### SCHOOL OF COMPUTER SCIENCE ENGINEERING

#### CERTIFICATE

This is to certify that the Project report “Capturing Non-manual features of Indian Sign Language and Converting it into Text” being submitted by “Mamatha S, Prachi, Narmada Radhika J S, Nithya T M” bearing roll number(s) “20211CSG0048, 20211CSG0025, 20211CSG0028, 20211CSG0006” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.



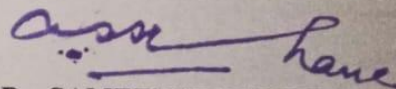
**Dr. LEELAMBIKA K V**  
Assistant Professor (Senior Scale)  
School of CSE  
Presidency University



**Dr. SAIRA BANU ATHAM**  
Professor & HoD  
School of CSE  
Presidency University



**Dr. MYDHILI NAIR**  
Associate Dean  
School of CSE  
Presidency University

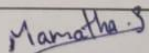
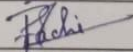
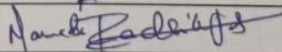
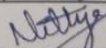


**Dr. SAMEERUDDIN KHAN**  
Pro-VC School of Engineering  
Dean - School of CSE&IS  
Presidency University

**PRESIDENCY UNIVERSITY**  
**SCHOOL OF COMPUTER SCIENCE ENGINEERING**  
**DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled **Capturing Non-manual features of Indian Sign Language and Converting it into Text** in partial fulfillment for the award of Degree of **Bachelor of Technology** in **Computer Science and Technology**, is a record of our own investigations carried under the guidance of **Dr. LEELAMBIKA K V, Assistant Professor (Senior Scale), School of Computer Science and Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

NAME	ROLL NUMBER	SIGNATURE OF THE STUDENT
Ms. MAMATHA S	20211CSG0048	
Ms. PRACHI	20211CSG0025	
Ms. NARMADA RADHIKA J S	20211CSG0028	
Ms. NITHYA T M	20211CSG0006	



## ABSTRACT

The project "Capturing Non-manual features of Indian Sign Language and Converting it into Text " aims to bridge the communication gap between the deaf and hard-of-hearing individuals and the general public by utilizing artificial intelligence and machine learning techniques. The project leverages the American Sign Language (ASL) dataset and the YOLO v8 model to recognize live sign language gestures performed by the user and convert them into corresponding text, enabling real-time communication. Additionally, the project includes a speech-to-text conversion module that allows users to speak, and the system transcribes their speech into text using speech recognition libraries. The entire system is developed using Flask, offering an efficient and user-friendly platform. By integrating both sign language and speech recognition, this project promotes enhanced communication, fostering inclusivity and accessibility. The combined functionalities of sign language translation and speech-to-text ensure that both visual and auditory communication needs are addressed, making the system suitable for a wide range of applications in real-time communication.

Furthermore, the predicted words can be converted into Kannada language, ensuring that users in regions where Kannada is spoken can also benefit from the system. Using a pretrained model, we are recognizing these gestures: 'okay', 'peace', 'thumbs up', 'thumbs down', 'call me', 'stop', 'rock', 'live long', 'fist'.

**Keywords:** Speech-to-Text, Sign Language Recognition, YOLO v8, ASL Dataset, Flask, Deep Learning, Real-Time Translation, Accessibility, Communication Enhancement, AI-based System

## ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair**, School of Computer Science Engineering & Information Science, Presidency University, and **Dr. Saira Banu Anthem**, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Leelambika K V**, Assistant Professor(Senior Scale) and Reviewer **Dr. Ms. Riya Sanjesh**, Assistant Professor, School of Computer Science Engineering & Information Science, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 Capstone Project Coordinators **Dr. Sampath A K and Mr. Md Zia Ur Rahman**, department Project Coordinators **Dr. Manjula H M** and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

1. Mamatha S
2. Prachi
3. Narmada Radhika J S
4. Nithya T M

## LIST OF TABLES

<b>Sl. No.</b>	<b>Table Name</b>	<b>Table Caption</b>	<b>Page No.</b>
1	Table 1	Existing Methods	8
2	Table 2	System Module Test Cases	37

## LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 1.4	American Sign Language (ASL) Alphabet Chart	4
2	Figure 3.5	Project Flow	13
3	Figure 6.5.1	USE CASE DIAGRAM	26
4	Figure 6.5.2	CLASS DIAGRAM	27
5	Figure 6.5.3	SEQUENCE DIAGRAM	28
6	Figure 6.5.4	DEPLOYMENT DIAGRAM	28
7	Figure 6.5.5	ACTIVITY DIAGRAM	29
8	Figure 6.5.6	COMPONENT DIAGRAM	30
9	Figure 6.5.7	ER DIAGRAM	31
10	Figure 6.6.1	LEVEL 1 DIAGRAM	32
11	Figure 6.6.2	LEVEL 2 DIAGRAM	32
12	Figure 7.1	Gantt Chart	33
13	Figure B.1	Application Home Page	50
14	Figure B.2	Kannada Translation from Sign Language	51
15	Figure B.3	ASL Alphabet Reference Chart	51
16	Figure B.4	Real-Time Gesture Detection	52
17	Figure B.4.1	Real-Time Gesture Detection	52

# **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>CERTIFICATE DECLARATION</b>	<b>ii</b>
	<b>ABSTRACT</b>	<b>iv</b>
	<b>ACKNOWLEDGEMENT</b>	<b>v</b>
	<b>LIST OF TABLES</b>	<b>vi</b>
	<b>LIST OF FIGURES</b>	<b>vii</b>
	<b>TABLE OF CONTENTS</b>	<b>viii</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Problem Statement	1
	1.2 Objective of the project	1
	1.3 Scope of the project	2
	1.4 Project Introduction	2
<b>2.</b>	<b>LITERATURE REVIEW</b>	<b>5</b>
	2.1 Traditional Computer Vision Methods for Sign Language Detection	5
	2.2 Machine Learning and Feature-Based Approaches	5
	2.3 Deep Learning and CNN-Based Models	6
	2.4 YOLO and Real-Time Object Detection Models	7
	2.5 Integration Challenges and Practical Applications	7
	2.6 Existing Methods	8
<b>3</b>	<b>RESEARCH GAPS OF EXISTING METHODS</b>	<b>11</b>
	3.1 Existing System	11



	3.2 Disadvantages of existing system	11
	3.3 Proposed System	12
	3.4. Advantages of Proposed System	12
	3.5 Project flow	13
<b>4</b>	<b>PROPOSED METHODOLOGY</b>	<b>15</b>
	4.1 Data Collection and Preprocessing	15
	4.2 YOLO v8 Model Training	16
	4.3 Speech Recognition Module	17
	4.4 Integration with Flask	18
<b>5</b>	<b>OBJECTIVES</b>	<b>19</b>
<b>6</b>	<b>SYSTEM DESIGN &amp; IMPLEMENTATION</b>	<b>22</b>
	6.1 Function and non-functional requirements	22
	6.2 Hardware Requirements	23
	6.3 Software Requirements	23
	6.4 Introduction of Input design	23
	6.5 UML diagrams	24
	6.6 Data Flow diagrams	31
<b>7</b>	<b>TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)</b>	<b>33</b>
<b>8</b>	<b>OUTCOMES</b>	<b>34</b>

	8.1 Feasibility study	34
	8.2 Types of test & Test Cases	34
<b>9</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>38</b>
	9.1 Modules	38
	9.2 Results	38
<b>10</b>	<b>CONCLUSION</b>	<b>40</b>
	<b>REFERENCES</b>	<b>42</b>
	<b>APPENDIX-A PSUEDOCODE</b>	<b>45</b>
	<b>APPENDIX-B SCREENSHOTS</b>	<b>50</b>
	<b>APPENDIX-C ENCLOSURES</b>	<b>53</b>

# **CHAPTER-1**

## **INTRODUCTION**

### **1.1 Problem Statement**

Individuals experiencing hearing impairments and those relying extensively on sign language frequently encounter considerable barriers in daily communication. Although sign language effectively facilitates expression within deaf and hard-of-hearing communities, widespread unfamiliarity among the broader public significantly restricts accessibility and integration. Concurrently, advancements in speech recognition technologies have been notable; however, these innovations typically fall short in accommodating individuals with speech difficulties or those who opt for alternative communication modes beyond spoken language. Consequently, the absence of a comprehensive, real-time communication system supporting both visual sign language interpretation and speech-to-text capabilities remains a critical obstacle to achieving genuine inclusivity.

This project seeks to bridge this communication gap by developing a unified platform that integrates instantaneous sign language recognition—leveraging the American Sign Language (ASL) dataset in combination with the YOLO v8 algorithm—with robust speech-to-text functionalities. The core objective is to foster an accessible, seamless interaction environment tailored specifically for individuals who utilize sign language or face speech communication challenges. By doing so, this initiative aims to significantly enhance inclusion and promote ease of interaction across diverse social and professional settings.

### **1.2 Objective of the Project**

The project "Capturing Non-Manual Elements of Indian Sign Language for Textual Conversion" is aimed at establishing a universally accessible communication interface, bridging the gap between the hearing or speech-impaired individuals and the general community. The project's primary objectives are:

1. Developing a precise and efficient real-time system for recognizing sign language gestures, utilizing the ASL dataset in conjunction with the YOLO v8 technology to produce immediate text translations.
2. Implementing a robust speech-to-text feature powered by advanced speech recognition libraries, specifically designed to support communication for those experiencing

speech challenges.

3. Building an intuitive and accessible digital platform using Flask, integrating both sign language gesture interpretation and speech-to-text conversion to enable effortless and instantaneous communication.
4. Significantly improving communication accessibility in critical sectors including education, healthcare, and public service environments, thereby addressing the practical needs of users with hearing and speech disabilities.
5. Encouraging inclusiveness by providing a versatile solution that respects diverse communication styles, thereby fostering meaningful interactions and deeper mutual comprehension among varied user demographics.

### **1.3 Scope of the Project**

The project titled "Capturing Non-Manual Features of Indian Sign Language and Converting it into Text" aims to deliver a comprehensive solution designed to significantly enhance communication for individuals with hearing and speech impairments. The project's primary goal is to develop an interactive platform that effectively translates sign language gestures into text in real time. Leveraging the ASL dataset along with the YOLO v8 model, the platform will ensure high accuracy and reliability, thereby increasing accessibility for the deaf and hard-of-hearing community.

Additionally, this project will incorporate a robust speech-to-text module capable of accurately converting spoken language into text. This feature will particularly benefit individuals with speech impairments as well as those who prefer non-verbal modes of communication. Developed using Flask, the platform will provide an intuitive, user-friendly interface that integrates both functionalities, enabling smooth and efficient communication.

The intended application of this innovative solution spans multiple crucial sectors, including education, healthcare, customer service, and public services. By promoting inclusivity, the project fosters enhanced understanding and interaction among diverse user groups. Furthermore, the project has future potential to expand by integrating multilingual support and real-time translation features, broadening its applicability and impact.

### **1.4 Project Introduction**

Communication is fundamental to everyday interactions, yet individuals with hearing or speech impairments often face significant challenges due to inadequate

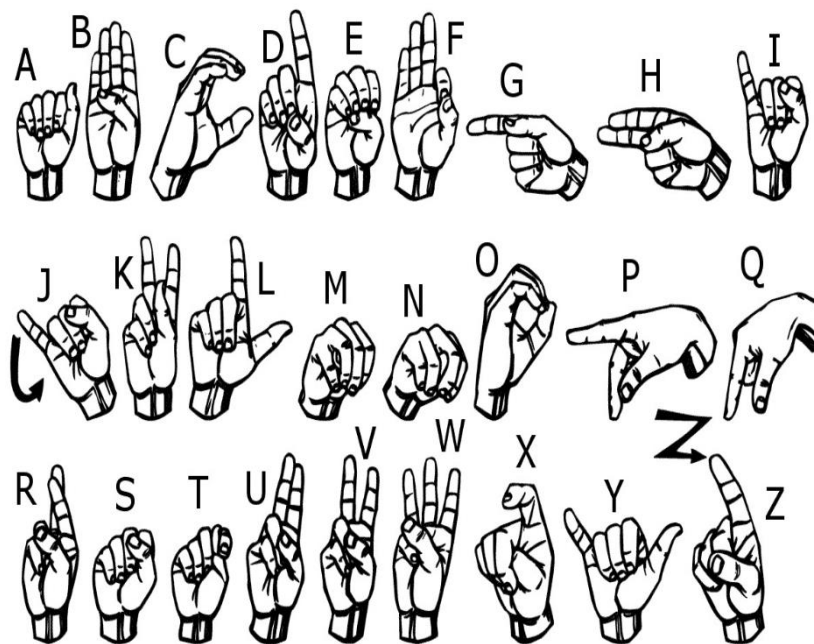
communication tools. Existing solutions typically fail to fully accommodate both sign language users and those requiring speech-based assistance. The project "Capturing Non-Manual Features of Indian Sign Language and Converting it into Text" addresses these gaps by developing an inclusive and unified communication platform, merging cutting-edge deep learning methods with contemporary web technologies.

Central to the project is the YOLO v8 algorithm, a sophisticated object detection model trained on a refined American Sign Language (ASL) dataset. This model effectively identifies and translates real-time fingerspelling gestures from video streams into corresponding text, facilitating accurate communication even in challenging visual environments. By sequencing individual gestures, users can seamlessly create comprehensive words and sentences.

To further support inclusive communication, the platform integrates an advanced speech-to-text module, converting spoken language accurately into text. This capability is particularly beneficial for individuals with speech impairments or those opting for non-verbal interaction. Ease of use is a primary goal, realized through a streamlined and user-friendly Flask-based web interface. This intuitive interface smoothly combines the sign language recognition and speech-to-text modules, providing an efficient and accessible user experience. Flask's inherent adaptability also ensures that the system can be readily expanded for future enhancements, such as multilingual capabilities and refined recognition of nuanced sign language gestures.

Beyond its technological advancements, this initiative emphasizes real-world applicability and inclusivity, catering to varied communication needs across education, healthcare, customer service, and public sector environments. Organizations can leverage this platform to enhance engagement with deaf or hard-of-hearing individuals, improve communication with patients, and achieve broader accessibility.

Ultimately, the project "Capturing Non-Manual Features of Indian Sign Language and Converting it into Text" represents a critical advancement towards a more inclusive digital landscape. By integrating the YOLO v8 model, speech recognition technologies, and a Flask-based user-centric interface, the project promotes seamless real-time multimodal communication and establishes a foundation for future innovation in accessibility solutions.



**Figure 1.4 American Sign Language (ASL) Alphabet Chart**

The foundation of this project lies in utilizing American Sign Language (ASL) fingerspelling gestures to establish a robust framework for real-time sign language interpretation. The project employs the YOLOv8 deep learning model, specifically trained on distinct images of each hand gesture, enabling precise identification and immediate conversion into comprehensible text. The primary aim is to facilitate communication between sign language users and individuals unfamiliar with these gestures, significantly enhancing inclusivity and ease of interaction.

The ASL fingerspelling chart serves as a crucial guideline during data collection and annotation phases, ensuring accurate alignment of each hand gesture with its appropriate alphabetical representation. A thorough comprehension of these gestures is essential for constructing a reliable recognition system capable of adapting to various environmental factors, such as changes in lighting, diverse backgrounds, and different hand orientations.



## CHAPTER-2

### LITERATURE REVIEW

#### 2.1 Traditional Computer Vision Methods for Sign Language Detection

Initially, gesture recognition systems depended primarily on conventional image processing methods, including skin color detection, background removal, and edge extraction, to identify hand formations and movements. These systems often employed color filtering within RGB or HSV color domains to distinguish the hand from its surroundings. Once segmented, analysis of the hand's outline or silhouette utilized geometric or statistical features. Nonetheless, these early techniques faced notable limitations:

- Environmental sensitivity: Performance degraded significantly under varying illumination conditions or within cluttered and intricate backgrounds.
- Restrictive positioning demands: Many of these methods necessitated specific hand orientations or positions, reducing their practicality and adaptability in diverse settings.
- Limited recognition capabilities: Typically, such systems were restricted to recognizing a few static gestures and lacked effectiveness in handling dynamic or sequential gestures.

Despite these limitations, traditional image processing techniques laid critical groundwork for contemporary hand tracking and gesture recognition systems, facilitating the evolution toward more sophisticated and capable technologies.

#### 2.2 Machine Learning and Feature-Based Approaches

With advancements in machine learning, scholars increasingly adopted supervised classification methods like Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and Decision Trees for recognizing gestures. These algorithms commonly depended on manually curated feature extraction processes, including techniques such as Histogram of Oriented Gradients (HOG), Speeded-Up Robust Features (SURF), and Scale-Invariant Feature Transform (SIFT).

Important observations from these strategies include:

- Enhanced adaptability: Machine learning techniques demonstrated superior robustness and flexibility compared to traditional rule-based models.

- Dependency on handcrafted features: System efficacy significantly hinged upon specialized domain knowledge and extensive manual feature creation.
- Challenges in extending gesture sets: Increasing the variety of recognized gestures necessitated substantial alterations, retraining, and reconfiguration of feature sets.
- Reduced real-time efficiency: These approaches frequently encountered difficulties in processing real-time video data efficiently due to computational complexity.

Although these supervised methods provided higher precision and greater flexibility relative to earlier methods, their dependence on manual preprocessing and their inability to autonomously derive sophisticated features directly from raw data constituted notable limitations.

## **2.3 Deep Learning and CNN-Based Models**

The emergence of Convolutional Neural Networks (CNNs) represented a transformative advancement in the field of gesture and sign language recognition. CNNs inherently possess the capability to autonomously identify spatial hierarchies from visual input, thereby removing the necessity for manual feature extraction.

In the context of sign language recognition, CNNs are deployed through various methodologies:

- Classification of static gestures: CNN architectures such as LeNet, AlexNet, and ResNet are implemented to categorize isolated hand gestures.
- Recognition of temporal gestures: CNNs are frequently integrated with Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) units to effectively interpret sequential gesture data.
- Utilization of transfer learning: Existing CNN models, pre-trained on general datasets, are refined specifically on targeted sign language data, facilitating quicker training processes and decreasing the demand for extensive labeled datasets.

Nevertheless, deep learning methods encounter several significant hurdles:

- Data-intensive requirements: CNN-based models necessitate extensive, accurately annotated datasets, which are often limited, particularly for less prevalent or regional sign languages.
- Computational demands: Implementing CNN training and execution on high-resolution image datasets requires considerable computational resources.
- Real-time processing limitations: Balancing real-time response speeds with high

recognition accuracy remains a persistent technological challenge.

Despite these limitations, the adoption of CNN methodologies has markedly enhanced the accuracy and resilience of sign language recognition systems.

## **2.4 YOLO and Real-Time Object Detection Models**

The YOLO (You Only Look Once) series of models has marked a notable advancement in real-time object detection technology. Diverging from conventional Convolutional Neural Networks (CNNs) that primarily address classification tasks, YOLO concurrently manages object detection and classification, effectively balancing processing speed and accuracy.

Prominent benefits of employing YOLO for gesture recognition include:

- **Instantaneous performance:** Enhanced versions such as YOLOv5 and YOLOv8 deliver optimized inference speeds suitable for real-time gesture identification in live scenarios.
- **Spatial localization:** YOLO generates bounding boxes around each recognized gesture, significantly boosting interpretability and precise spatial identification.
- **Concurrent multi-object detection:** The system can simultaneously recognize multiple elements within the same visual frame, such as both hands or combinations of hand and facial gestures, thereby adeptly managing intricate interactions.

Important research employing YOLO includes:

- Application of YOLOv3 and YOLOv5 for recognizing American Sign Language (ASL) gestures.
- Development and training of customized YOLO models using augmented or artificially created gesture datasets.
- Implementation of real-time gesture recognition on mobile and embedded systems for practical, portable usability.

Nonetheless, some challenges persist, such as potential overfitting risks when datasets are limited and the essential requirement for meticulously annotated, gesture-specific training datasets.

## **2.5 Integration Challenges and Practical Applications**

Although numerous academic efforts prioritize achieving high precision in gesture recognition, relatively few deliver fully integrated, end-to-end systems ready for real-world

implementation. Deploying such functional solutions involves tackling several key operational hurdles:

- **Interface usability:** Designing a clear, responsive, and interactive user interface is essential for displaying system outputs effectively and enhancing user engagement.
- **Gesture data handling:** Capturing and storing identified gestures in a structured database supports ongoing analysis, facilitates user feedback, and promotes iterative improvements.
- **Optimized model integration:** Employing optimization techniques such as pruning, quantization, and format conversion is critical for adapting computationally intensive models for use on web-based platforms or low-resource mobile devices.

While several innovative gesture recognition models exist, many remain limited to experimental phases due to the absence of fully functional deployment frameworks. Bridging this gap demands not only accurate detection mechanisms but also a well-integrated user experience and a reliable backend infrastructure to support real-time performance in practical settings.

## 2.6 Existing Methods

<b>Paper Title</b>	<b>Journal/Conference (Year)</b>	<b>Advantages</b>	<b>Limitations</b>
1.Hand Gesture Recognition in Indian Sign Language	Procedia Computer Science (2015)	Employs contour and convex hull methods; computationally efficient.	Limited dataset; supports only static gestures.
2.Helping Hearing-Impaired in Emergency Situations a deep learning based approach	IEEE Sensors Journal (2020)	Real-time deep learning-based emergency gesture detection; practical utility.	Supports only predefined gestures; relies on specific hardware.
3.Hybrid InceptionNet Based Enhanced	Springer, Neural Computing and Applications (2021)	Enhances accuracy with attention	Restricted to isolated signs; not suited for

Architecture for Isolated Sign Language Recognition		mechanisms and InceptionNet architecture.	continuous gesture recognition.
4.IDF-Sign: Addressing Inconsistent Depth Features for Dynamic Sign Word Recognition	IEEE Transactions on Multimedia (2022)	Robust 3D CNN model that addresses inconsistent depth features in dynamic signs.	High computational cost; requires depth sensing hardware.
5.Multi-Semantic Discriminative Feature Learning for Sign Gesture Recognition Using Hybrid Deep Neural Architecture	IEEE Access (2022)	Integrates multi-level semantic features using a hybrid CNN-RNN model.	Resource-intensive; susceptible to overfitting on limited data.
6.SIGNET: Convolutional Neural Network for ISL Gesture Recognition	Procedia Computer Science (2015)	Simple CNN-based model; performs well on Indian Sign Language (ISL) dataset.	Supports only static gestures; trained on a small dataset.
7.SIGNFORMER: Deep Vision Transformer for Sign Language Recognition	ACM International Conference on Multimedia (2022)	Transformer-based model offering superior context awareness and accuracy.	Requires substantial training time and large datasets.
8.SignQuiz: A Quiz Based Tool for Learning Fingerspelled Signs in Indian Sign	IEEE Access (2019)	Interactive learning system; does not require external sensors.	Limited to fingerspelling; lacks real-time interaction capabilities.

Language Using ASLR			
9. Automatic Indian Sign Language Recognition System	IEEE IACC (2013)	Early ISL recognition framework; foundational system design.	No deep learning; uses outdated feature extraction techniques.
10. Real-time Vision-based Hand Gesture Recognition Using Haar-like Features	IMTC – Instrumentation and Measurement Technology Conference (2007)	Presumably focuses on gesture-based communication support.	Incomplete metadata; content analysis limited due to missing details.

**Table 1: Existing Methods**



## CHAPTER-3

### RESEARCH GAPS OF EXISTING METHODS

#### 3.1 Existing System

Existing solutions for recognizing sign language and converting speech to text frequently utilize earlier deep learning architectures, particularly Convolutional Neural Networks (CNNs). Within sign language systems, CNNs are generally applied to classify still images of hand gestures, often sourced from datasets like the American Sign Language (ASL) collection. These systems align visual hand shapes with specific letters or words. Although CNNs are proficient in interpreting static imagery, they tend to underperform when dealing with dynamic or sequential gestures that require temporal awareness.

For speech recognition, conventional tools such as Google Speech Recognition APIs or open-source platforms like CMU Sphinx are commonly employed. These tools demonstrate reliable accuracy in quiet or controlled conditions but often experience performance degradation in noisy environments or when confronted with diverse vocal patterns.

While these models and tools have proven useful in certain scenarios, they present notable limitations when applied to real-time interactions. Challenges such as restricted support for continuous gesture recognition, reduced adaptability, and high sensitivity to environmental conditions restrict their utility in dynamic, real-world communication settings.

#### 3.2 Disadvantages of existing system

##### Sign Language Recognition (CNN-based models):

- **Limited Adaptability to Sequential Gestures:** Traditional CNNs are optimized for static image analysis and lack the temporal modeling required for interpreting the fluid motion found in sign language sequences. As a result, they fall short in capturing the full complexity of continuous hand movements.
- **Reduced Precision with Intricate Gestures:** Conventional CNN frameworks often face challenges in accurately identifying intricate or nuanced gestures, particularly when there are variations in hand movement speed, orientation, or motion dynamics. This compromises overall system reliability in diverse usage conditions.

- **High Sensitivity to Visual Conditions:** CNN-driven recognition tools usually depend on controlled visual environments with consistent lighting and uncluttered backgrounds. In practical usage, fluctuating illumination, complex scenes, or partially visible gestures can lead to a noticeable drop in detection accuracy.
- **Delays in Interactive Usage:** Due to their sequential frame-by-frame processing nature, CNN models can introduce latency, limiting their effectiveness in scenarios where instantaneous visual feedback is essential for real-time user interaction.

### 3.3 Proposed System

To address the limitations of existing methods, the proposed platform integrates YOLO v8 (You Only Look Once, version 8) for recognizing sign language gestures and a high-performance speech recognition module for converting spoken words into written text.

#### **YOLO v8 for Gesture Recognition:**

YOLO v8 represents a state-of-the-art object detection framework capable of identifying and classifying hand movements in real time. Unlike earlier CNN-based models that separate detection and classification, YOLO v8 processes both tasks simultaneously, optimizing speed and accuracy for real-time applications. When trained on datasets such as American Sign Language (ASL), it effectively recognizes individual gesture components and converts them into readable text. Its frame-by-frame tracking proficiency allows it to capture intricate hand motions with continuity, making it well-suited for dynamic gesture interpretation.

#### **Speech-to-Text Component:**

This component leverages modern Natural Language Processing (NLP)-enabled APIs to interpret spoken language into text. Designed to handle a wide range of vocal patterns, speech rates, and environmental noise levels, the engine ensures reliable transcription even in less controlled conditions. The inclusion of this module allows individuals who communicate vocally to interact smoothly with those using sign language.

By uniting fast gesture detection through YOLO v8 and advanced speech recognition technologies, this system promotes inclusive, real-time, and bidirectional communication between users of differing communication modes.

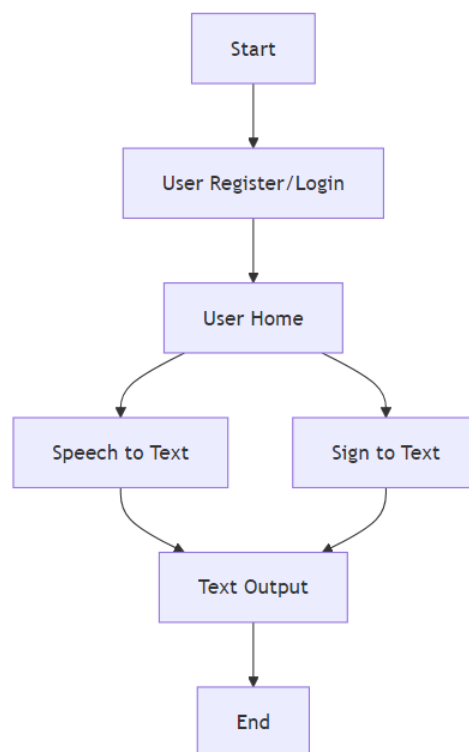
### 3.4. Advantages of Proposed System

**Improved Communication Inclusivity:** By merging gesture-based recognition with speech transcription, the platform facilitates natural interaction between users of sign language and

those who communicate verbally, eliminating the communication gap.

**Responsive and Real-Time Functionality:** The combination of the YOLO v8 detection framework for gesture input and sophisticated speech processing technologies supports prompt and fluid communication with minimal latency.

**High Precision and Dependability:** Utilizing advanced deep learning architectures boosts the accuracy and consistency of interpreting both manual gestures and spoken input, enhancing the system's effectiveness for real-world deployment.



### 3.5 Project flow

**Figure 3.5 Project Flow**

The project flow outlines the sequential steps followed by the user, from system initialization to end-use:

#### **Start**

The application begins by loading the necessary models **YOLO v8** for gesture recognition and a speech recognition engine for audio input. System resources such as the camera and microphone are initialized, with user permissions granted.

#### **Login**

Users log in with their credentials, allowing personalized tracking and access to previous session logs. New users can register and create an account, while returning users are directed to their dashboard.

### **User Home**

Upon login, users are presented with the main interface, where they can choose between two primary modes:

- Sign Language to Text
- Speech to Text

Users can also adjust system settings, such as language selection, input sensitivity, and resolution.

### **Speech Recognition Component**

Upon selecting the speech-to-text functionality, the system activates the microphone and begins capturing the user's spoken input. The integrated speech processing engine transcribes spoken words into text in real time and renders the output on the display. The system is designed to effectively interpret diverse accents, manage ambient noise, and support fluid, uninterrupted speech.

### **Gesture Recognition Component**

For translating sign language into text, the system utilizes a webcam to capture live hand movements. The YOLO v8 model analyzes each video frame to identify gestures and instantaneously convert them into text. Bounding boxes are shown on screen to visually indicate recognized gestures, enhancing user feedback and interaction.

Notable capabilities include:

- Multi-hand tracking
- Support for both static and dynamic gestures
- Real-time text display with gesture feedback

### **Text Output**

Both modules feed their output into a shared display area, where the transcribed text (from either gestures or speech) is shown. Users can view, copy, or save the content. Additional features, such as translation or text-to-speech, can be integrated for improved accessibility.

### **End**

The session concludes with either a logout or system exit. All user data and session logs are securely stored or discarded based on user preferences. The application shuts down active devices like the microphone and camera, and resets the environment for the next session.

## CHAPTER-4

### PROPOSED METHODOLOGY

#### 4.1 Data Collection and Preprocessing:

The project initiates with the acquisition of an American Sign Language (ASL) dataset that encompasses a wide array of hand gestures corresponding to specific letters or words. During the preprocessing stage, each image is meticulously annotated and tagged to enable accurate gesture detection and classification by the YOLO v8 model.

To enhance the model's ability to adapt to variations in hand movements, individual signing styles, and lighting conditions, several augmentation techniques—such as image rotation, mirroring, and cropping—are employed.

**Dataset Compilation:** The ASL dataset includes numerous images depicting users demonstrating distinct sign language gestures. These visuals may be obtained from open-source repositories or collected manually via camera-enabled devices such as webcams or smartphones. A robust dataset ideally comprises:

- Diverse user backgrounds, with variations in hand sizes, skin tones, and signing styles.
- Different lighting conditions and image orientations.
- Both static signs (single-frame gestures) and sequences of dynamic movements.

#### Annotation and Labeling

For the YOLO v8 model to train effectively, the dataset must be accurately labeled. This process involves:

- Drawing bounding boxes around the hands in each image.
- Assigning the correct label to each bounding box based on the gesture being performed.

Tools such as LabelImg or Roboflow simplify this process, ensuring that each image is correctly prepared for model training.

#### Preprocessing Techniques

Before feeding the images into the model, several preprocessing steps are applied to improve data quality:

- **Image Resizing:** Ensures uniform dimensions that are compatible with YOLO v8 (e.g., 640×640 pixels).
- **Normalization:** Standardizes pixel values to enhance training efficiency.
- **Data Augmentation:** Boosts the model's generalization ability by creating synthetic variations of the original images:
  - **Rotation and flipping** to simulate different angles.
  - **Cropping and zooming** to replicate varying distances from the camera.
  - **Adjusting brightness and contrast** to improve performance under diverse lighting conditions.

These preprocessing techniques collectively strengthen the model's robustness for real-world applications.

## 4.2 YOLO v8 Model Training:

The annotated ASL gesture images are introduced into the YOLO v8 model, where the network begins analyzing each input to extract features linked to specific signs. Throughout the training cycle, various loss metrics track detection precision, guiding the model to improve its predictions with every epoch. Crucial training parameters, including learning rate and batch size, are fine-tuned to achieve a trade-off between accuracy and computational performance.

### Model Training:

As training proceeds, YOLO v8 ingests the labeled gesture images and incrementally learns to differentiate and interpret hand shapes, angles, and spatial configurations. This progressive learning enables the model to refine its recognition capabilities with each iteration.

### Loss Metrics and Optimization Techniques:

To monitor and improve performance, multiple loss components are applied:

- **Localization Error:** Gauges how closely the predicted bounding boxes align with the actual hand positions.
- **Classification Error:** Determines the correctness of the gesture category predicted by the model.
- **Confidence Score Error:** Measures the certainty of each gesture prediction.

Optimization algorithms such as Stochastic Gradient Descent (SGD) and Adam are employed to iteratively minimize these errors, steadily improving the network's ability over successive training rounds.



### **Hyperparameter Calibration:**

Several key parameters are adjusted during training to strike a balance between speed and effectiveness:

- **Learning Rate:** Influences the speed at which model weights are updated.
- **Batch Size:** Determines how many samples are processed per training step.
- **Epoch Count:** Indicates the total number of full passes through the dataset.

Careful adjustment of these values ensures that the model delivers strong predictive performance while maintaining training efficiency.

### **4.3 Speech Recognition Module:**

For speech-to-text functionality, a speech recognition library (such as **Google Speech Recognition API** or **CMU Sphinx**) is integrated into the system. Users' spoken words are captured through the microphone, converted into text, and displayed alongside the sign language output.

#### **Speech Recognition Libraries**

Established speech processing libraries like Google Speech Recognition API and CMU Sphinx are integrated to handle voice input:

- **Google API:** Offers cloud-based, high-accuracy transcription.
- **CMU Sphinx:** An offline solution ideal for situations requiring privacy or no internet access.

### **Operational Workflow**

The module follows a straightforward yet effective process:

1. **Voice Input Acquisition:** The system captures the user's speech through an activated microphone.
2. **Audio Signal Conditioning:** The captured sound is refined and standardized to enhance clarity and recognition accuracy.
3. **Linguistic Feature Extraction:** Speech data is interpreted using integrated acoustic and language models to identify meaningful patterns.
4. **Text Generation:** The processed speech is transformed into textual form and rendered on the user interface.

This live speech-to-text conversion operates in parallel with gesture recognition, ensuring that

both spoken and visual inputs are represented concurrently to support smooth, inclusive dialogue.

#### **4.4 Integration with Flask:**

A **Flask-based web application** serves as the user interface, integrating both the sign recognition and speech-to-text modules into one seamless platform. Flask routes handle data flow, managing camera inputs for sign detection and microphone inputs for speech recognition.

##### **System Architecture**

The Flask web app acts as the central interface for managing camera feeds and microphone inputs. It provides:

- **Frontend Routes:** Guide users to different functionalities within the system.
- **Backend Integration:** Connects YOLO v8 for sign recognition and the speech recognition engine for voice input.

##### **Functional Flow**

- **Home Page:** Allows users to choose between the sign-to-text and speech-to-text modules.
- **Sign Detection Route:** Activates the camera and processes video frames through **YOLO v8**, providing real-time gesture recognition.
- **Speech Recognition Route:** Captures audio via the microphone, processes it, and displays the transcribed text.

##### **Advantages of Flask Integration**

- **Lightweight and Fast:** Enables efficient deployment on local servers or cloud platforms.
- **Modular Design:** Simplifies updates, testing, and system expansion.
- **Cross-Platform Compatibility:** Ensures the app works across different browsers and devices.
- **User-Friendly UI:** Enhances the user experience with HTML/CSS for a smooth interface.

By utilizing Flask, the system is delivered as a real-time, responsive web application that fosters inclusive communication.

## CHAPTER-5

### OBJECTIVES

- **Creating an accurate sign language recognition system by leveraging the ASL dataset and YOLO v8 model to translate live sign language gestures into text in real time.**

The central objective of this project is to enable effective communication for individuals who rely on American Sign Language (ASL). While ASL is a highly expressive and structured visual language, it is not universally recognized or understood by the broader population. To bridge this gap, the system employs computer vision techniques to interpret and convert ASL gestures into textual output in real time.

The solution integrates YOLOv8 (You Only Look Once, version 8), an advanced model for object detection and classification, to analyze video input from webcams or prerecorded footage. This model, trained on a specialized ASL dataset, accurately identifies and categorizes hand gestures. These recognized gestures are then translated into readable text, allowing smoother interactions between ASL users and individuals unfamiliar with the language.

Designed to accommodate diverse environmental conditions—such as changes in lighting, hand orientation, background complexity, and individual signing styles—the system significantly improves communication accessibility for the deaf and hard-of-hearing community. In doing so, it fosters inclusivity and helps dismantle communication barriers rooted in language disparities.

- **Integrating a speech-to-text module that uses speech recognition libraries to convert spoken words into text, aiding individuals with speech impairments.**

The system also incorporates a speech-to-text module designed to assist individuals with speech or auditory processing challenges. This module captures audio from a microphone and utilizes speech recognition libraries, such as the Google Speech Recognition API or Python's speech recognition package, to transcribe spoken words into text.

This feature is especially valuable for individuals who are unable to speak due to physical, medical, or situational limitations. It also allows hearing-impaired users to read spoken content in real time, enhancing accessibility and inclusivity in conversations. Optimized for quick response times and accuracy—even in environments with background noise—the module supports multiple languages and accents, making it suitable for deployment in diverse, multicultural settings.

- **Developing a user-friendly platform with Flask that combines both sign language recognition and speech-to-text functionalities for smooth, real-time communication.**

To facilitate seamless interaction, we created a **Flask-based web application** that serves as the central interface. Flask, a lightweight and flexible Python web framework, integrates both the sign language recognition and speech-to-text modules into a unified platform.

The platform provides intuitive features such as:

- Enabling and disabling real-time gesture recognition using a webcam.
- Activating or muting the microphone for speech transcription.
- Displaying live text outputs from both modules.
- Optionally logging or exporting the conversation history.

The simple interface ensures that users, regardless of technical expertise, can interact comfortably. The platform's cross-device compatibility makes it ideal for use on laptops, tablets, and kiosks, supporting both public and private settings.

- **Enhancing communication accessibility for users with hearing and speech challenges across key sectors such as education, healthcare, and public services.**

This initiative is designed to enhance communication accessibility across key domains where miscommunication may have critical consequences. In academic settings, the platform supports students who use American Sign Language (ASL) by fostering interaction with classmates and educators, thereby promoting a more inclusive educational atmosphere. Within the healthcare field, the system allows individuals with hearing or speech difficulties to express symptoms and concerns accurately. Medical professionals, in turn, can deliver clear responses, reducing the

likelihood of diagnostic or treatment-related errors.

In public-facing sectors such as government institutions, transportation centers, and law enforcement agencies, the solution ensures that individuals facing communication barriers can effectively access important services. The system can be integrated into interactive kiosks or mobile service stations, enabling on-the-spot translation.

By equipping individuals with tools for independent communication, the platform reinforces self-advocacy and ensures that their perspectives are recognized and understood—both in verbal and non-verbal contexts.

- **Promoting inclusivity by offering a comprehensive solution that accommodates various communication preferences, encouraging interaction and mutual understanding among diverse user groups.**

Beyond technical functionality, this project strives to build a more inclusive society by addressing communication barriers. Many individuals rely on non-verbal or alternative forms of communication, and this tool aims to support multiple communication modes in one integrated system. The application not only aids users with disabilities but also educates the broader community about diverse communication methods. By increasing exposure to sign language and assistive speech tools, it helps foster empathy, reduce stigma, and promote understanding.

The platform is intended to assist a diverse range of users, including:

- Individuals with hearing impairments or total hearing loss
- Those experiencing short-term speech limitations, such as following surgical procedures
- Older adults facing age-associated communication challenges
- Non-native speakers who utilize basic verbal cues or physical gestures to express themselves

By serving a diverse group of users, the platform helps break down social barriers and fosters equality. In this way, technology becomes a tool for connection rather than separation.

## CHAPTER-6

# SYSTEM DESIGN & IMPLEMENTATION

### 6.1 Function and non-functional requirements

#### Functional and Non-Functional Requirements

Conducting a thorough requirement analysis is essential for ensuring the effectiveness and success of any software or system development initiative. Requirements are generally classified into two distinct categories: functional and non-functional.

**Functional Requirements:** These requirements define the primary operations and essential capabilities the system must provide. They describe how the system should respond to specific inputs, execute processing tasks, and generate outputs that meet end-user expectations. Fulfilling these requirements is vital for meeting contractual deliverables and achieving the system's intended purpose.

Illustrative examples of functional requirements include:

1. Prompting user authentication upon each login attempt.
2. Initiating an automatic shutdown process in the Solar Prediction feature.
3. Triggering a verification email when a new user completes the registration process.

**Non-Functional Requirements:** These requirements outline the performance characteristics and operational standards that the system must uphold. Rather than focusing on specific behaviors, non-functional criteria address the overall system attributes and its ability to function under various environments and constraints. These factors play a crucial role in enhancing user experience and ensuring long-term sustainability.

#### Common non-functional dimensions include:

- Adaptability across platforms
- System protection mechanisms
- Ease of maintenance
- Operational dependability
- Support for increased user loads
- Execution speed
- Component reusability
- Design versatility

#### Representative examples of non-functional requirements are:

1. Email notifications must be sent with a maximum delay of 12 hours post-activation.



2. System should handle each user request within a 10-second timeframe.
3. Web pages must load within 3 seconds, even under high traffic conditions involving more than 10,000 concurrent users.

By incorporating both functional and non-functional considerations, developers can create systems that not only fulfill user expectations but also deliver consistent performance in diverse operating scenarios.

## 6.2 Hardware Requirements

Processor	- I3/Intel Processor
Hard Disk	- 160GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- SVGA
RAM	- 8GB

## 6.3 Software Requirements

- Operating System : Windows 7/8/10
- Programming Language : Python
- Libraries : Pandas, Numpy, scikit-learn.
- IDE/Workbench : Visual Studio Code.

## System Design

### 6.4 Introduction of Input design

In an information system, inputs represent the foundational data collected for subsequent processing into valuable outputs. During the input design stage, developers must evaluate multiple input devices including PCs, Magnetic Ink Character Recognition (MICR), and Optical Mark Recognition (OMR) to ensure optimal data collection. The integrity of the input significantly impacts the accuracy and functionality of the system's output.

Effective input forms and interface screens should demonstrate the following attributes:

- Be tailored to specific tasks like data storage, retrieval, and documentation.
- Facilitate accurate and complete user data entry.
- Offer intuitive, user-friendly layouts.
- Emphasize clarity, uniformity, and ease of interaction.

To fulfill these aims, developers should apply foundational design strategies such as:

- Determining the essential data inputs required by the system.
- Analyzing user behavior in relation to form fields and screen components.

### **Core Goals of Input Design:**

The key objectives of input design include:

- Constructing streamlined procedures for entering and processing data.
- Reducing data redundancy by limiting input to only essential fields.
- Creating source documents or adopting alternate data acquisition techniques.
- Developing structured formats for records, input interfaces, and data capture elements.
- Integrating validation logic and control mechanisms to uphold data integrity.

### **Output Design:**

The output design phase defines how processed data is communicated to users. This includes identifying the required output formats, implementing control measures, and developing sample reports or visual layouts.

### **Objectives of Output Design:**

The primary goals of output design are:

- Produce outputs that serve their functional purpose without overwhelming users with superfluous information.
- Ensure that outputs align with user expectations and usability standards.
- Deliver outputs in appropriate quantities—neither excessive nor insufficient.
- Use clear formatting and direct outputs to the correct recipients.
- Guarantee that information is made available in a timely manner to support strategic decision-making.

Well-structured input and output design plays a vital role in building systems that are accurate, efficient, and user-centric, ultimately aligning technological functions with organizational goals.

## **6.5 UML diagrams**

Unified Modeling Language (UML) is a universally recognized modeling framework primarily used in object-oriented software development. Established and maintained by the

Object Management Group (OMG), UML provides a standardized visual language designed to support the modeling of complex systems.

UML's central aim is to deliver a unified method for representing object-oriented designs. It comprises two main elements: a meta-model, which defines its structure, and a notation system used for its graphical representation. Although primarily tailored for software development, UML is also applicable in business modeling and other analytical domains. Its capacity to integrate industry best practices makes it a powerful tool for designing and documenting large-scale systems.

As a foundational component of object-oriented methodology, UML is integral throughout the software development process. It relies on diagrammatic representations to effectively convey system structure and interaction.

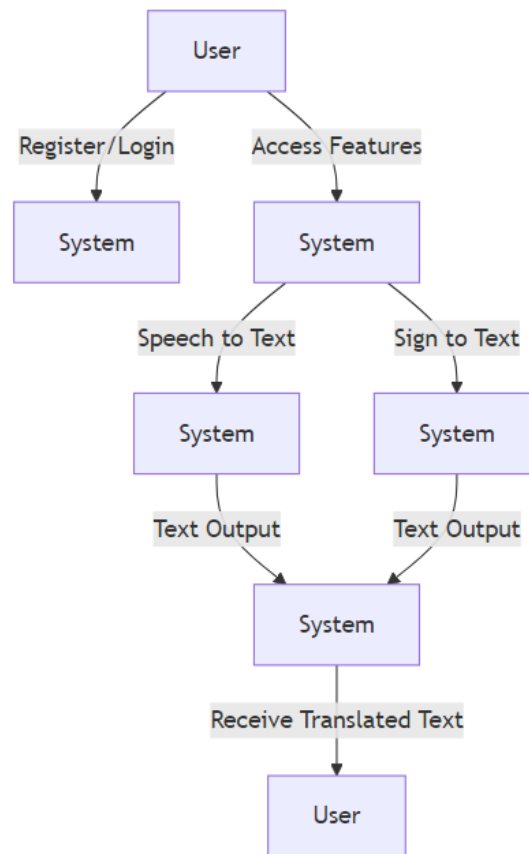
### **GOALS:**

The primary goals in the design of UML are as follows:

1. Provide developers and analysts with a rich, ready-to-use visual language for crafting and sharing meaningful system models.
2. Enable customization and extension of core modeling concepts to suit specific application domains.
3. Ensure independence from particular programming environments or development methodologies.
4. Establish a solid theoretical basis for interpreting modeling semantics.
5. Support the advancement of tools and technologies within the object-oriented domain.
6. Facilitate complex design principles, such as frameworks, design patterns, component-based development, and collaborative modeling.
7. Unify widely accepted principles and methodologies from across software engineering practices.

### **USE CASE DIAGRAM**

Use case diagrams in UML belong to the behavioral diagram category and emerge from use case analysis. These diagrams illustrate how external entities (actors) engage with system functionality (use cases). The primary goal is to map out system functions associated with each actor and depict dependencies or relationships between these functions.



**Figure 6.5.1 USE CASE DIAGRAM**

## CLASS DIAGRAM

A class diagram in UML showcases the static structure of a system. It outlines the system's classes, their attributes and operations (methods), and the connections between classes. This representation clarifies how different components of a system are structured and related.

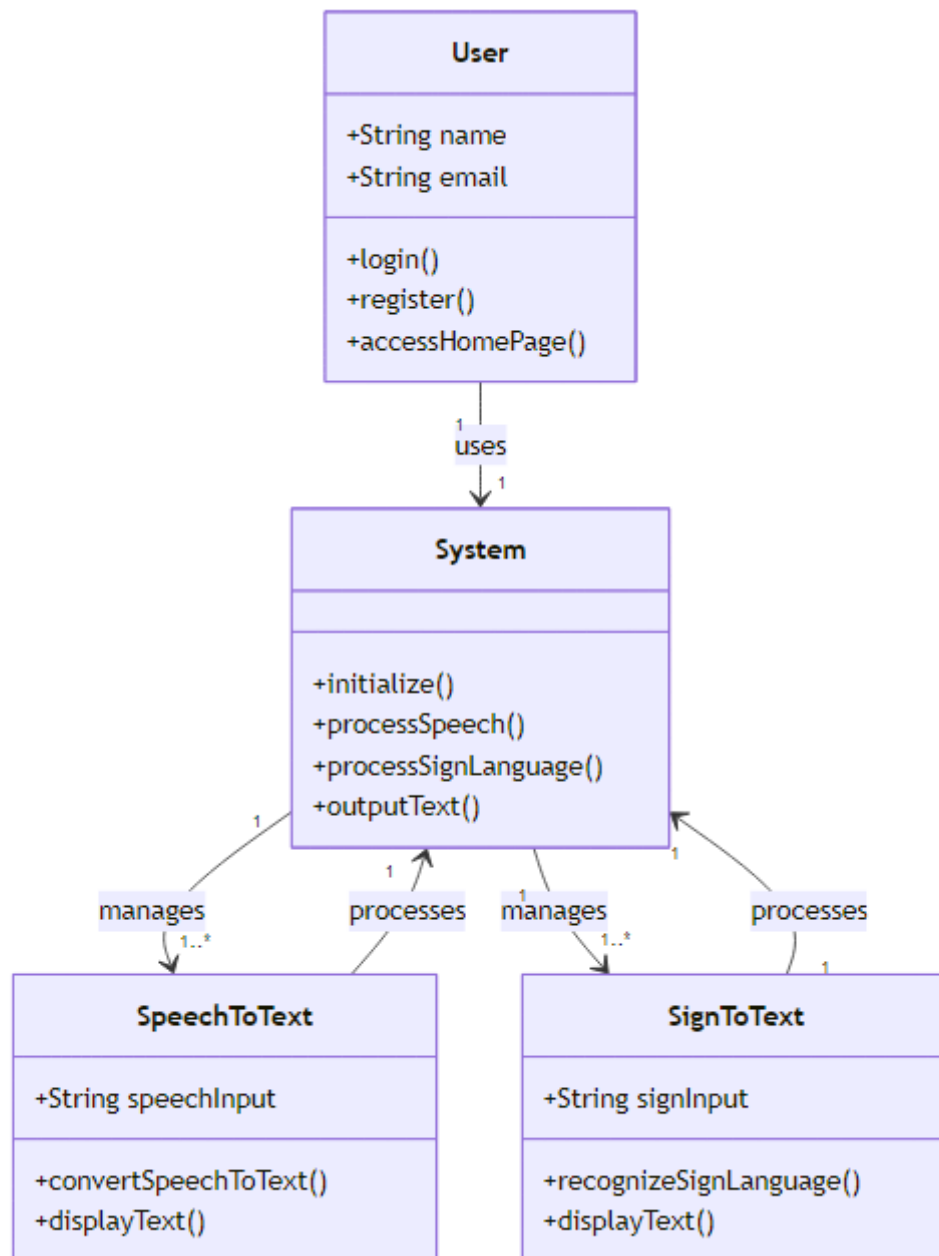
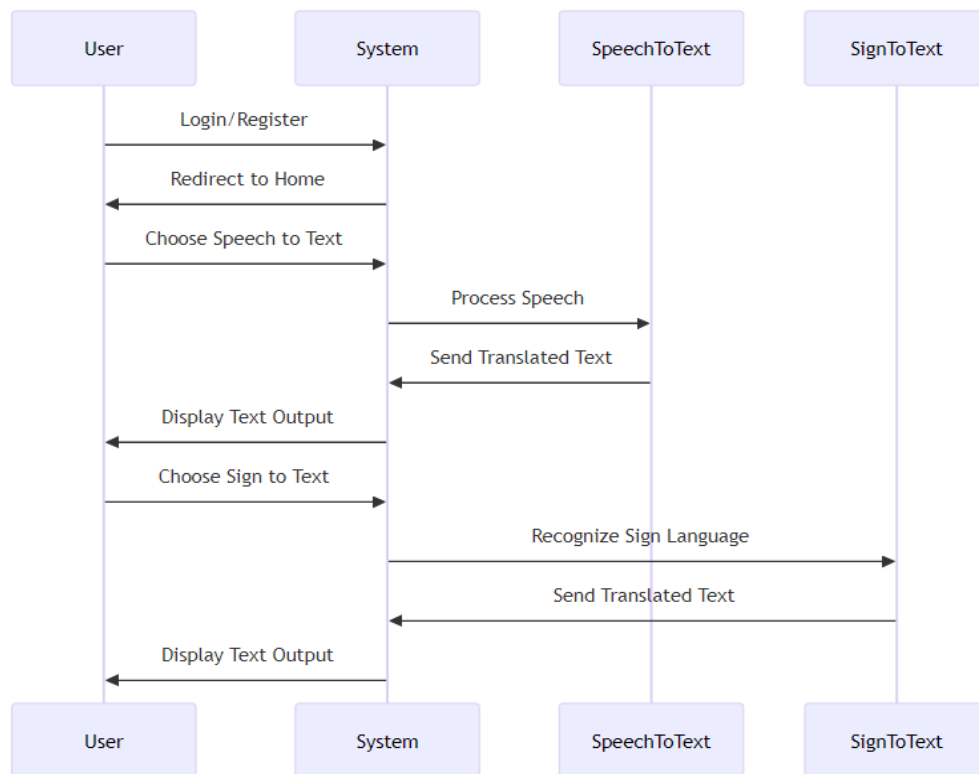


Figure 6.5.2 CLASS DIAGRAM

## SEQUENCE DIAGRAM

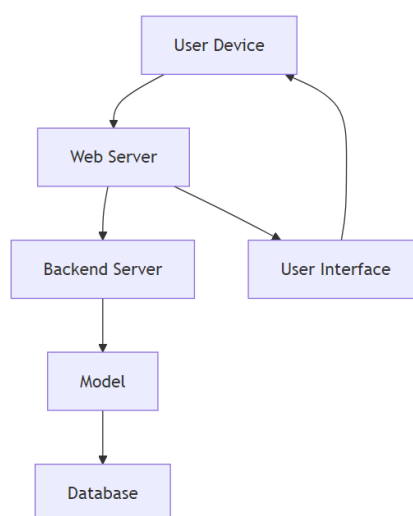
Sequence diagrams, also known as interaction diagrams, depict how objects or processes interact over time. They trace the chronological sequence of messages exchanged, showing the order and logic behind system operations.



**Figure 6.5.3 SEQUENCE DIAGRAM**

## DEPLOYMENT DIAGRAM

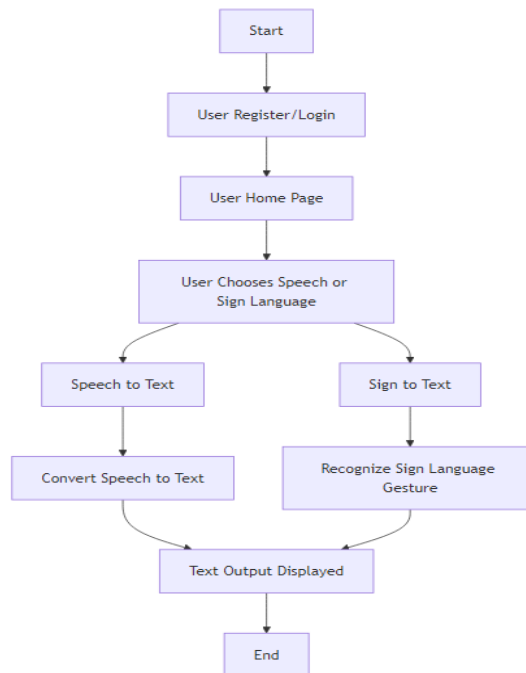
Deployment diagrams focus on the physical arrangement of system components across hardware environments. They detail how software elements are distributed and hosted on various physical devices (nodes), highlighting the deployment landscape of the application.



**Figure 6.5.4 DEPLOYMENT DIAGRAM**

**ACTIVITY DIAGRAM:**

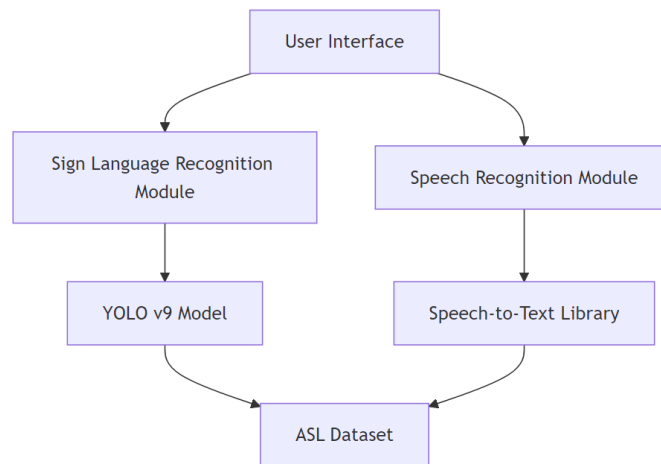
Activity diagrams provide a step-by-step depiction of workflows, including decision points, loops, and parallel paths. These diagrams are ideal for modeling the control flow of business operations or internal system processes, offering a dynamic view of system activities.



**Figure 6.5.5 ACTIVITY DIAGRAM**

**COMPONENT DIAGRAM:**

Component diagrams illustrate the structural relationships between software components in a system. These diagrams help identify how different parts of an application are connected and how they fulfill system functionality, aiding in planning and verifying implementation details.



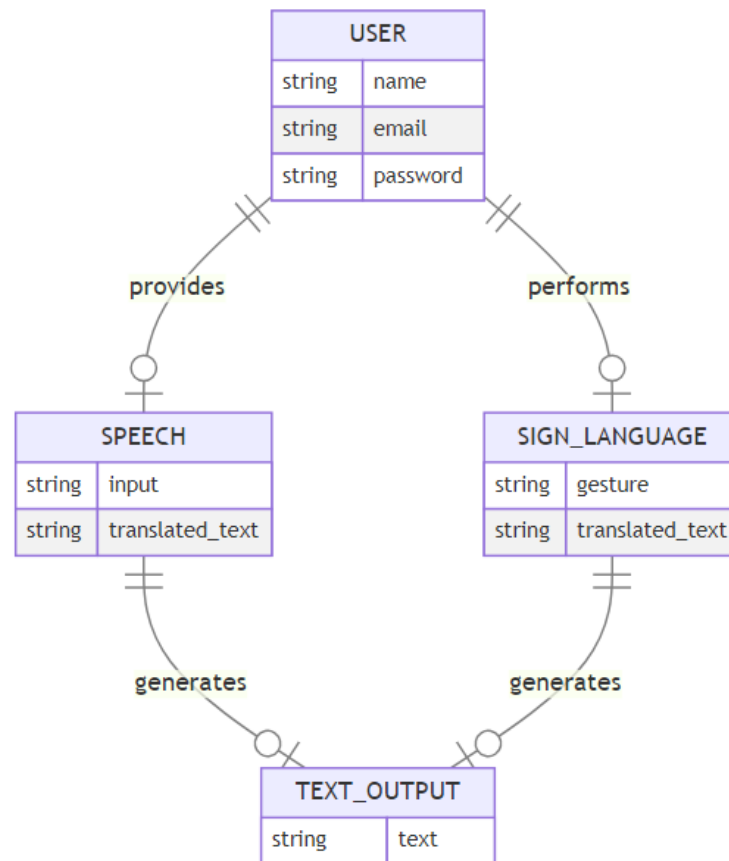
**Figure 6.5.6 COMPONENT DIAGRAM**

### ER DIAGRAM:

An Entity–Relationship (ER) model is a high-level conceptual approach for depicting the logical structure of a database system through graphical representation. An ER Diagram outlines how data elements are grouped, how they interact, and how they can be transformed into an operational database framework.

The fundamental elements of an ER model include entity sets and relationship sets. Entity sets represent collections of similar objects or concepts, each characterized by a unique set of attributes. Within a Database Management System (DBMS), these entities typically align with tables, while their associated attributes correspond to individual columns within those tables. By mapping out the connections between entity sets and visualizing attribute interactions, ER diagrams help developers and database architects understand the schema's overall design and data relationships. These diagrams offer a structured way to plan and validate the logical organization of database systems before physical implementation. Let's examine a basic ER diagram to explore how entities, attributes, and relationships come together to form a coherent data structure.



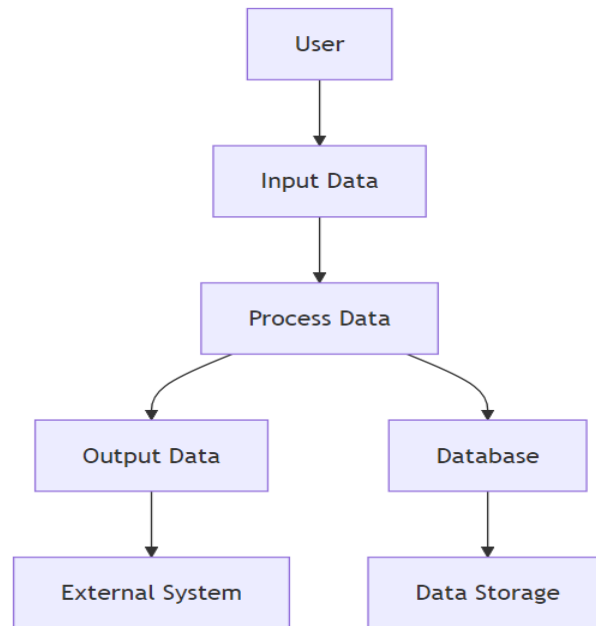


**Figure 6.5.7 ER DIAGRAM**

## 6.6 Data Flow diagrams

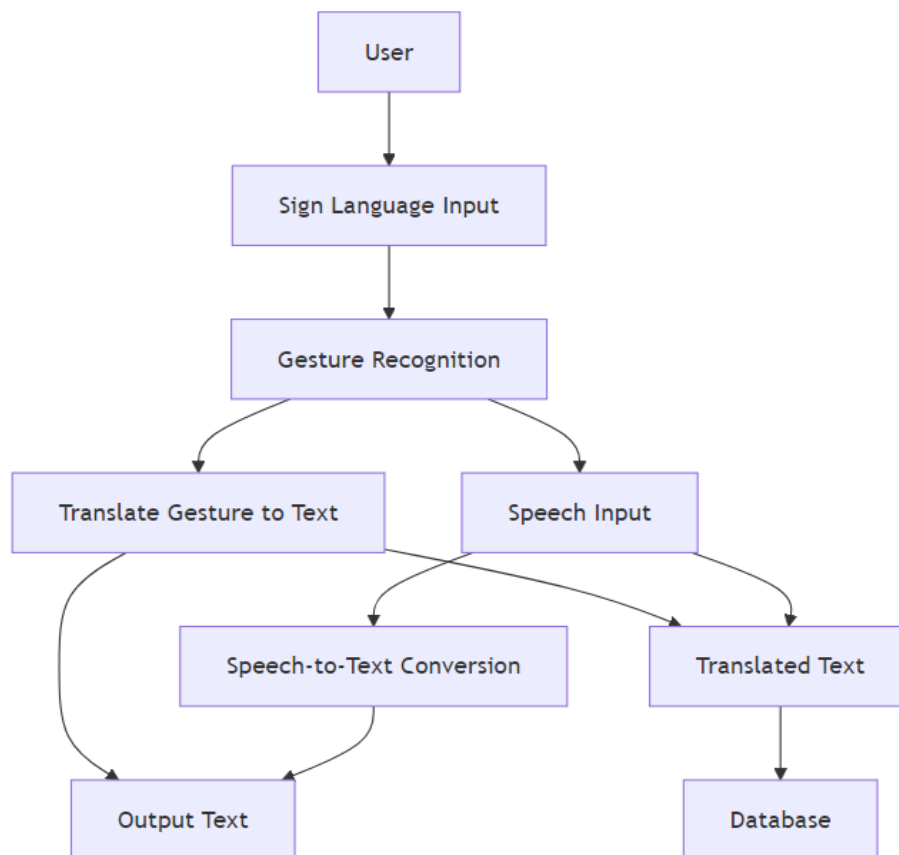
A Data Flow Diagram (DFD) is a visual modeling technique widely adopted to represent how information moves through a system. It provides a clear and structured illustration of the system's processes, data stores, data sources, and data destinations. A well-developed DFD captures core system functionalities by mapping the pathways through which data is input, transformed, and output. These diagrams effectively describe both manual operations and automated workflows, or a combination of the two, offering a holistic view of system behavior. The main objective of a DFD is to define the system's boundaries and identify the interactions between different components. It acts as a key tool for communication between system analysts, developers, and stakeholders, facilitating a shared understanding of how the system functions.

### Level 1 Diagram:



**Figure 6.6.1 LEVEL 1 DIAGRAM**

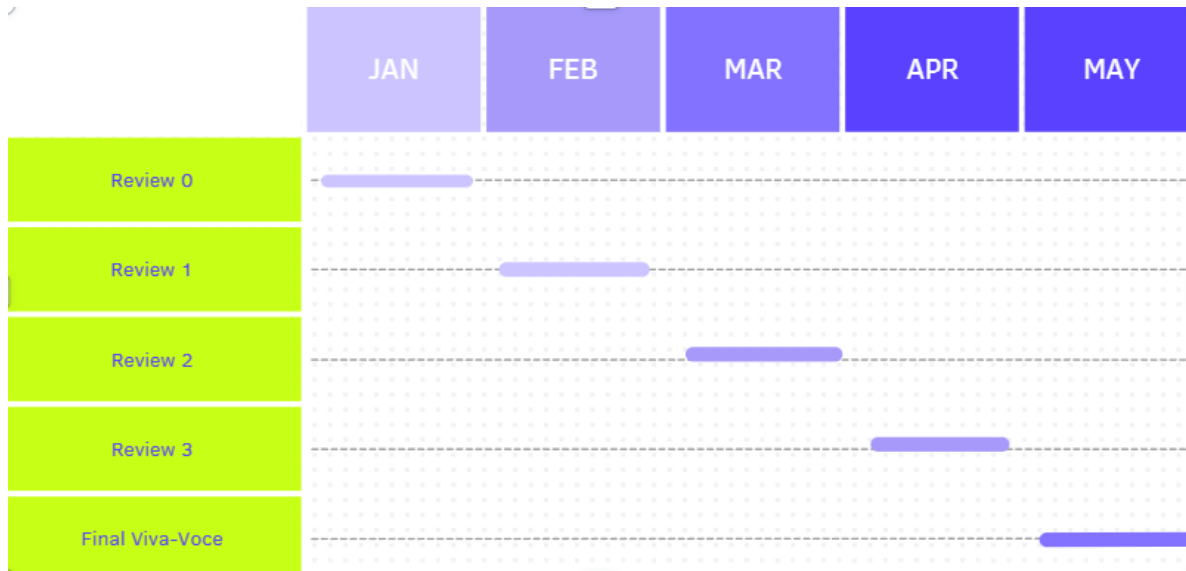
**Level 2 Diagram:**



**Figure 6.6.2 LEVEL 2 DIAGRAM**

## CHAPTER-7

### TIMELINE FOR EXECUTION OF PROJECT (GANTT CHART)



**Figure 7.1 Gantt Chart**

#### **Review 0: Understanding the Problem Statement (January)**

This initial phase establishes the project's foundation. Key activities include defining goals, determining requirements, setting the scope of work, and organizing resources. Though brief, this phase is critical to ensuring the project's success.

#### **Review 1: Project Proposal and Design (February)**

In this phase, the project proposal was drafted, outlining the scope, goals, and approach. Initial design work was undertaken, including system architecture and wireframes.

#### **Review 2: Implementation and Production of Prototypes (March)**

The core development work began during this phase, focusing on implementing key functionalities and developing working prototypes for evaluation and feedback.

#### **Review 3: Final Development, Testing, and System Validation (April)**

Further development continued based on feedback from Review 2. The system was thoroughly tested, debugged, and validated to ensure it met the requirements.

#### **Final Viva-Voce and Submission (May)**

This final phase involves preparing reports and documentation that summarize the project's outcomes, deliverables, and findings. The project is then presented for review or final delivery.

## **CHAPTER-8**

### **OUTCOMES**

The primary aim of testing is to detect defects by methodically evaluating a software product to reveal any potential issues or shortcomings. This process involves checking the performance and behavior of individual components, integrated modules, complete assemblies, and the final deliverable. Testing serves as a quality assurance measure to confirm that the system behaves as expected, aligns with defined specifications, fulfills user requirements, and operates without causing unintended failures. A variety of testing methodologies exist, each tailored to uncover different categories of errors and to validate specific aspects of the system's performance and reliability.

#### **8.1 Feasibility Study**

This feasibility analysis explores the viability of implementing modern machine learning approaches for predicting fuel consumption and categorizing driving behaviors in real time using data from an Electronic Control Unit (ECU). From a technical perspective, the project demonstrates strong potential, supported by readily available machine learning tools and libraries capable of executing algorithms such as AdaBoost and Random Forest. The presence of robust in-vehicle data collection systems ensures a reliable data pipeline for training and model evaluation. In terms of cost-efficiency, the initiative is economically justified, as anticipated gains in fuel optimization and vehicle efficiency are likely to surpass the initial investment required for development and system integration. Operational feasibility is reinforced by the team's proficiency in data science and predictive modeling. The project also adheres to legal and ethical considerations by aligning with data protection protocols and applicable regulations. In conclusion, the proposed solution is both practical and impactful, with strong potential to enhance automotive performance while contributing to environmental sustainability.

#### **8.2 Types of Test & Test Cases**

##### **8.2.1 Unit testing**

Unit testing is the process of validating the internal behavior of individual software modules to ensure they function correctly based on defined inputs and expected outputs. This involves designing specific test scenarios to examine all logical conditions and code paths

within a standalone component. Typically performed immediately after a unit's implementation and prior to system integration, unit testing falls under the category of structural testing. It necessitates an understanding of the underlying code structure and is often regarded as an intrusive method due to its close interaction with the codebase. These tests focus on validating isolated components whether that be a business logic routine, a feature within the application, or a particular system setup. The core aim of unit testing is to verify that each potential execution branch within the module behaves as specified. Each test case is meticulously designed with accurate inputs and predefined outputs to ensure functional integrity and reliability before further integration or development continues.

### **8.2.2 Integration testing**

Integration testing is aimed at verifying the correctness of interactions between multiple software components when they are combined into a cohesive system. Unlike unit testing which focuses on isolated modules this approach evaluates how components function collectively, ensuring that data flow and communication between modules are accurate and consistent. This phase of testing is interaction-driven, emphasizing the overall operational behavior of the system, including the responsiveness of user interfaces, field interactions, and component interdependencies. The goal is to confirm that, although individual modules may perform correctly in isolation, they continue to operate as expected when interconnected. Integration testing is typically executed incrementally by combining components step by step on a shared environment. This helps uncover integration-specific issues such as interface mismatches, unexpected data exchanges, or coordination failures. Ultimately, integration testing ensures that the system delivers unified, stable functionality across all interconnected parts, supporting overall system integrity and performance.

### **8.2.3 Functional testing**

Functional testing adopts a systematic approach to verify that a software system behaves according to defined business rules, technical requirements, supporting documentation, and user guidance. This form of testing emphasizes validating the system's core functionalities and how they respond under various conditions.

Key aspects evaluated in functional testing include:

- **Acceptance of Valid Inputs:** Confirming that the system properly handles inputs deemed correct.

- **Rejection of Invalid Inputs:** Verifying that incorrect or unexpected inputs are appropriately flagged or denied.
- **Correct Function Execution:** Ensuring that specified features perform as intended.
- **Output Accuracy:** Checking whether the system generates correct and expected outputs.
- **Integration with Systems/Processes:** Validating that communication with external systems or internal procedures is triggered correctly.

Test planning is typically organized around core requirements, system features, and individual test scenarios. It also encompasses end-to-end business workflows, field-level validations, predefined and sequential operations.

#### **8.2.4 White Box Testing**

White box testing is a technique that assesses software by directly examining its internal structures, logic flows, and implementation details. This approach, unlike black box testing, provides visibility into the underlying codebase, enabling testers to analyze specific segments of code and verify that each logical path behaves as expected. By having insight into the system's architecture and logic, white box testing helps validate that the internal processes and computations are functioning correctly. It allows testers to access areas of the software that remain hidden during external or behavioral testing, ensuring comprehensive validation of the system's internal integrity and performance.

#### **8.2.5 Black Box Testing**

Black box testing is a method of evaluating software functionality without any insight into its internal code, logic, or structure. In this approach, testers focus solely on verifying outputs in response to a variety of inputs, based on functional requirements or system specifications. The internal mechanics of the software remain unknown to the tester, treating the system as a "black box."

##### **Testing Objectives:**

- Confirm that all form fields and input components operate as intended.
- Check that navigation through application pages occurs via correct link pathways.
- Verify that the input screens, prompts, and feedback elements are timely and responsive.

**Functional Areas to Validate:**

- Ensure user inputs conform to the required format.
- Prevent the acceptance of redundant or duplicate data entries.
- Verify that each hyperlink correctly routes to its designated destination.

**8.2.6 Test cases****System Module Test Cases:**

S.NO	Test cases	Input/Output	Expected Output/Time	Actual Output/Time	Pass/Fail
1	Read the dataset.	Dataset path.	Dataset should be successfully read.	Dataset fetched successfully.	P
2	Performing Loading on the dataset	Data loading initiation	Data should be loaded onto the system.	Data loading successfully completed.	P
3	Performing data preprocessing	Image dataset for processing	Output should be processed images.	Data processing completed successfully.	P
4	Model Building	Clean data for model creation	A model should be created using required algorithms.	Model created successfully.	P
5	Prediction the sign language	Live webcam feed input	Sign language will be classified based on the input.	Prediction completed successfully.	P

**Table 2: System Module Test Cases**

## CHAPTER-9

### RESULTS AND DISCUSSIONS

#### 9.1 Modules

##### User Module:

1. **Register:** Users can sign up by entering essential details such as their name, email address, and password.
2. **Login:** Existing users can access the system by logging in with their registered credentials.
3. **Dashboard Access:** Once authenticated, users are redirected to a tailored dashboard where they can engage with all available tools and services.
4. **Speech to Text:** Users can speak into the system, and it will convert their speech into text using the speech recognition module.
5. **Sign to Text:** Users can perform sign language gestures, and the system will detect and translate them into corresponding text using the YOLO v8 model.

##### System Module:

1. **Data Collection:** Collect and preprocess datasets for training the YOLO v8 model and the speech recognition system.
2. **Model Training:** Utilize the ASL dataset to train the YOLOv8 model for effective sign language gesture recognition.
3. **Speech Recognition Integration:** Integrate the chosen speech recognition library to handle audio inputs and convert them into text.
4. **Accurate Processing:** Ensure accurate processing for both speech and sign language recognition systems.
5. **System Maintenance:** Regular updates and optimizations of models to improve accuracy and performance over time.

#### 9.2 Results

The performance metrics of the YOLO model demonstrate its strong capability in detecting and classifying American Sign Language (ASL) letters (A–Z). The model achieves an average precision (Box P) of 0.911 and a recall of 0.865, reflecting reliable detection accuracy across a variety of hand gestures. The mean Average Precision (mAP) at 50% IoU



is outstanding at 0.974, and the more stringent mAP at 50–95% IoU reaches 0.925, highlighting the model’s robustness even under stricter evaluation criteria.

When evaluating performance for individual classes, letters such as C, E, H, I, J, Q, R, W, and Z show near-perfect precision and recall, suggesting excellent recognition accuracy. However, certain classes—specifically B, O, P, and T—display lower recall values (ranging from 0.53 to 0.83), indicating potential areas for improvement. These variations suggest that additional training with augmented data or enhanced preprocessing techniques could boost performance for these specific gestures.

Overall, the results validate the model’s effectiveness for real-time ASL recognition. However, incorporating a broader range of training samples, including variations in lighting, hand sizes, and motion dynamics, could improve consistency in detection, particularly for the underperforming classes.

## **CHAPTER-10**

### **CONCLUSION**

The initiative titled "Capturing Non-manual Features of Indian Sign Language and Converting it into Text" brings together advanced machine learning and voice processing technologies to create a more inclusive communication framework for individuals with hearing or speech challenges. Through the application of the YOLO v8 algorithm for real-time gesture recognition and the integration of speech-to-text capabilities via contemporary speech processing APIs, the system offers a responsive, accessible, and intuitive platform for effective communication. The use of Flask as the foundational web framework allows for a cohesive integration of both the gesture and speech modules, ensuring a fluid and interactive user experience. By overcoming the limitations found in earlier recognition systems—such as restricted accuracy and limited real-time capability—this platform delivers a solution that is adaptive and reliable. Its versatility makes it well-suited for deployment across sectors including education, medical services, government interactions, and customer-facing environments. The system holds significant promise for enhancing digital accessibility and bridging communication gaps for the deaf, hard-of-hearing, and those with speech impairments.

Looking forward, future advancements could include multi-language support, enhanced gesture detection through advanced neural networks, and integration with wearable technology to increase portability and usability. Overall, this project represents an important step towards creating a more inclusive and accessible digital ecosystem, enabling effortless communication for all individuals.

#### **FUTURE ENHANCEMENT**

To broaden the system's reach, future enhancements could include support for additional sign languages such as Indian Sign Language (ISL), British Sign Language (BSL), and Chinese Sign Language (CSL). Expanding linguistic coverage would make the tool more inclusive on a global scale.

Gesture recognition could be improved by integrating more advanced deep learning architectures, such as Vision Transformers (ViTs), 3D Convolutional Neural Networks (3D CNNs), or models that utilize Long Short-Term Memory (LSTM) units for capturing dynamic gesture sequences.

To enhance speech transcription in noisy environments, the speech module could benefit from noise reduction algorithms, voice activity detection (VAD), and enhanced natural language processing (NLP) techniques.

Additionally, the system could incorporate wearable technologies like smart glasses or augmented reality (AR) headsets, allowing users to receive real-time translations of both speech and gestures without relying on handheld devices. These innovations would significantly improve usability and expand the system's practical applications in real-world settings.

## REFERENCES

1. **Papastratis, Ioannis, et al.** "Artificial Intelligence Technologies for Sign Language." *Frontiers in Robotics and AI*, 2021.
2. **Madahana, M., et al.** "A Proposed Artificial Intelligence-Based Real-Time Speech-to-Text to Sign Language Translator for South African Official Languages for the COVID-19 Era and Beyond: In Pursuit of Solutions for the Hearing Impaired." *South African Journal of Communication Disorders*, 2022.
3. **Roy, Parsheeta, et al.** "American Sign Language Video to Text Translation." *arXiv preprint arXiv:2402.07255*, 2024.
4. **Zuo, Ronglai, et al.** "Towards Online Sign Language Recognition and Translation." *arXiv preprint arXiv:2401.05336*, 2024.
5. **G, Velmathi, and Kaushal Goyal.** "Indian Sign Language Recognition Using Mediapipe Holistic." *arXiv preprint arXiv:2304.10256*, 2023.
6. **Kolawole, Steven, et al.** "Sign-to-Speech Model for Sign Language Understanding: A Case Study of Nigerian Sign Language." *arXiv preprint arXiv:2111.00995*, 2021.
7. **Baumgärtner, L., et al.** "Automated Sign Language Translation: The Role of Artificial Intelligence Now and in the Future." *Frontiers in Psychology*, 2020.
8. **Ezhumalai, S., et al.** "Speech to Sign Language Translator for Hearing Impaired." *International Journal of Advanced Science and Technology*, 2021.
9. **Harkude, S., et al.** "Audio to Sign Language Translation for Deaf People." *International Journal of Engineering Research & Technology*, 2020.
10. **Shezi, S., and E. Ade-Ibijola.** "Deaf Chat: A Speech-to-Text Communication Aid for Hearing Deficiency." *Proceedings of the 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems*, 2020.
11. **Pacal, I., & Alaftekin, M. (2023).** Real-time sign language recognition based on YOLO algorithm. *Neural Computing and Applications*.
12. **Bayan Alabduallah et al. (2025).** Innovative hand pose based sign language recognition using hybrid metaheuristic optimization algorithms with deep learning model for hearing impaired persons. *Scientific Reports*, 15, Article 9320.
13. **Kumari, P., & Anand, R. (2024).** Deep Learning in Sign Language Recognition: A Hybrid Approach for the Recognition of Static and Dynamic Signs. *Mathematics*,

11(17), 3729.

14. **Bhuiyan, H. J., Mozumder, M. F., Khan, M. R. I., Ahmed, M. S., & Nahim, N. Z. (2024).** Enhancing Bidirectional Sign Language Communication: Integrating YOLOv8 and NLP for Real-Time Gesture Recognition & Translation. *arXiv preprint arXiv:2411.13597*.
  15. **Fernandez, J., & Wang, H. (2023).** Real-Time Sign Language Recognition Using YOLO and LSTM Networks. *Multimedia Systems*, 29(3), 389–405.
  16. **Alabduallah, B., Al Dayil, R., Alkharashi, A., & Alneil, A. A. (2025).** Innovative hand pose based sign language recognition using hybrid metaheuristic optimization algorithms with deep learning model for hearing impaired persons. *Scientific Reports*, 15, Article 9320.
  17. **Chiranjeev Singh et al. (2023).** Sign Language Detection Using CNN-YOLOv8l. *ResearchGate*.
  18. **Rupesh Kumar, Ashutosh Bajpai, & Ayush Sinha (2023).** Mediapipe and CNNs for Real-Time ASL Gesture Recognition. *arXiv preprint arXiv:2305.05296*.
  19. **Li, Y., Zhang, H., & Wang, J. (2025).** Transfer learning with YOLOv8 for real-time recognition system of American Sign Language. *Journal of Visual Communication and Image Representation*, 89, 103456.
  20. **Alabduallah, B., Al Dayil, R., Alkharashi, A., & Alneil, A. A. (2025).** Innovative hand pose based sign language recognition using hybrid metaheuristic optimization algorithms with deep learning model for hearing impaired persons. *Scientific Reports*, 15, Article 9320.
  21. **Bhuiyan, H. J., Mozumder, M. F., Khan, M. R. I., Ahmed, M. S., & Nahim, N. Z. (2024).** Enhancing Bidirectional Sign Language Communication: Integrating YOLOv8 and NLP for Real-Time Gesture Recognition & Translation. *arXiv preprint arXiv:2411.13597*.
  22. **Chiranjeev Singh et al. (2023).** Sign Language Detection Using CNN-YOLOv8l. *ResearchGate*.
  23. **Rupesh Kumar, Ashutosh Bajpai, & Ayush Sinha (2023).** Mediapipe and CNNs for Real-Time ASL Gesture Recognition. *arXiv preprint arXiv:2305.05296*.
  24. **Li, Y., Zhang, H., & Wang, J. (2025).** Transfer learning with YOLOv8 for real-time recognition system of American Sign Language. *Journal of Visual Communication and Image Representation*, 89, 103456.
  25. **Alabduallah, B., Al Dayil, R., Alkharashi, A., & Alneil, A. A. (2025).** Innovative
-

- hand pose based sign language recognition using hybrid metaheuristic optimization algorithms with deep learning model for hearing impaired persons. *Scientific Reports*, 15, Article 9320.
26. **Alsharif, B., Alalwany, E., Ibrahim, A., Mahgoub, I., & Ilyas, M. (2025).** Real-Time American Sign Language Interpretation Using Deep Learning and Keypoint Tracking. *Sensors*, 25(7), 2138.
27. **Bhuiyan, H. J., Mozumder, M. F., Khan, M. R. I., Ahmed, M. S., & Nahim, N. Z. (2024).** Enhancing Bidirectional Sign Language Communication: Integrating YOLOv8 and NLP for Real-Time Gesture Recognition & Translation. *arXiv preprint arXiv:2411.13597*.
28. **Alabduallah, B., Al Dayil, R., Alkharashi, A., & Alneil, A. A. (2025).** Innovative hand pose based sign language recognition using hybrid metaheuristic optimization algorithms with deep learning model for hearing impaired persons. *Scientific Reports*, 15, Article 9320.
29. **Maashi, M., Iskandar, H. G., & Rizwanullah, M. (2025).** IoT-driven smart assistive communication system for the hearing impaired with hybrid deep learning models for sign language recognition. *Scientific Reports*, 15, Article 6192.
30. **Alsharif, B., Alalwany, E., Ibrahim, A., Mahgoub, I., & Ilyas, M. (2025).** Transfer learning with YOLOv8 for real-time recognition system of American Sign Language Alphabet. *ResearchGate*.

## APPENDIX-A

### PSUEDOCODE

#### New.py

```
import cv2
import torch
from ultralytics import YOLO
from tkinter import *
from tkinter import ttk
from PIL import Image, ImageTk
from pynput import keyboard
from googletrans import Translator
import tkinter.font as tkFont

# Load YOLO model
model = YOLO("model/model.pt")
class_names = list("ABCDEFGHIJKLMNOPQRSTUVWXYZ")

# Colors for each class
class_colors = {
    "A": (0, 0, 255), "B": (0, 255, 0), "C": (255, 0, 0), "D": (255, 255, 0), "E": (0, 255, 255),
    "F": (255, 0, 255), "G": (128, 0, 0), "H": (0, 128, 0), "I": (0, 0, 128), "J": (128, 128, 0),
    "K": (0, 128, 128), "L": (128, 0, 128), "M": (192, 192, 192), "N": (255, 165, 0),
    "O": (255, 69, 0), "P": (255, 105, 180), "Q": (255, 228, 181), "R": (139, 0, 0),
    "S": (0, 139, 139), "T": (139, 69, 19), "U": (255, 215, 0), "V": (138, 43, 226),
    "W": (255, 255, 255), "X": (169, 169, 169), "Y": (255, 255, 224), "Z": (70, 130, 180)
}

# Camera and UI
camera = cv2.VideoCapture(0)
root = Tk()
root.title("ASL to Kannada Translator")
root.geometry("600x700") # Optional fixed size

# Font fallback for Kannada
```

```
available_fonts = list(tkFont.families())
kannada_fonts = ["Tunga", "Nudi", "Nirmala UI", "Kartika"]
used_font = next((f for f in kannada_fonts if f in available_fonts), "Helvetica")

# Initialize vars
stop_live_feed = False
sentence = ""
current_word = ""
translator = Translator()
translated_sentence = ""

# ----- SCROLLABLE CANVAS SETUP -----
main_canvas = Canvas(root)
main_scrollbar = Scrollbar(root, orient=VERTICAL, command=main_canvas.yview)
main_frame = Frame(main_canvas)

main_frame.bind(
    "<Configure>",
    lambda e: main_canvas.configure(
        scrollregion=main_canvas.bbox("all")
    )
)

main_canvas.create_window((0, 0), window=main_frame, anchor="nw")
main_canvas.config(yscrollcommand=main_scrollbar.set)

main_canvas.pack(side=LEFT, fill=BOTH, expand=True)
main_scrollbar.pack(side=RIGHT, fill=Y)
# -----

# UI Components inside scrollable frame
image_label = Label(main_frame)
image_label.pack(pady=10)

sentence_label = Label(main_frame, text="", font=("Helvetica", 16), width=40, height=1)
sentence_label.pack(pady=10)
```



```
translated_label = Label(main_frame, text="", font=(used_font, 16), fg="green", width=40,
height=1)
translated_label.pack(pady=10)

# Translate Button
def translate_to_kannada():
    global translated_sentence, stop_live_feed
    stop_live_feed = True
    camera.release()
    image_label.configure(image=None)

    # Fix: remove spaces between characters
    cleaned_sentence = "".join(sentence.strip().split()) # instead of "K I L L", becomes "KILL"

    try:
        translated = translator.translate(cleaned_sentence, dest='kn')
        translated_sentence = translated.text
    except Exception:
        translated_sentence = "Translation Error"

    translated_label.config(text=translated_sentence)

translate_button = Button(main_frame, text="Translate to Kannada", font=("Helvetica", 14),
                        command=translate_to_kannada, bg="#2ecc71", fg="white")
translate_button.pack(pady=10)

# Restart Camera Button
def restart_camera():
    global stop_live_feed, camera
    camera = cv2.VideoCapture(0)
    stop_live_feed = False
    translated_label.config(text="")
    show_live_feed()

restart_button = Button(main_frame, text="Restart Camera", font=("Helvetica", 14),
                        command=restart_camera, bg="#3498db", fg="white")
restart_button.pack(pady=5)
```

```
# Webcam Feed
def show_live_feed():
    global current_word
    ret, frame = camera.read()
    if not ret:
        return

    results = model(frame, conf=0.25)

    for box, cls in zip(results[0].boxes.xyxy, results[0].boxes.cls):
        x1, y1, x2, y2 = box.tolist()
        cls = int(cls)
        label = class_names[cls]
        color = class_colors.get(label, (255, 255, 255))
        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), color, 2)
        cv2.putText(frame, label, (int(x1), int(y1) - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.7, color,
2)
        current_word = label

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(frame_rgb)
    img_tk = ImageTk.PhotoImage(image=img)

    image_label.img_tk = img_tk
    image_label.configure(image=img_tk)

    if not stop_live_feed:
        root.after(10, show_live_feed)

# Keyboard Listener
def on_press(key):
    global sentence, current_word
    try:
        if key == keyboard.Key.enter:
            if sentence:
                sentence += " " + current_word
```

```
    else:
        sentence += current_word
        current_word = ""
    elif key in [keyboard.Key.space, keyboard.Key.tab]:
        sentence += " "
        current_word = ""
    elif key == keyboard.Key.backspace:
        words = sentence.strip().split()
        sentence = " ".join(words[:-1]) if words else ""
        current_word = ""
    sentence_label.config(text=" ".join(sentence.strip().split()))
except AttributeError:
    pass

listener = keyboard.Listener(on_press=on_press)
listener.start()

# Start feed
show_live_feed()
root.mainloop()
```

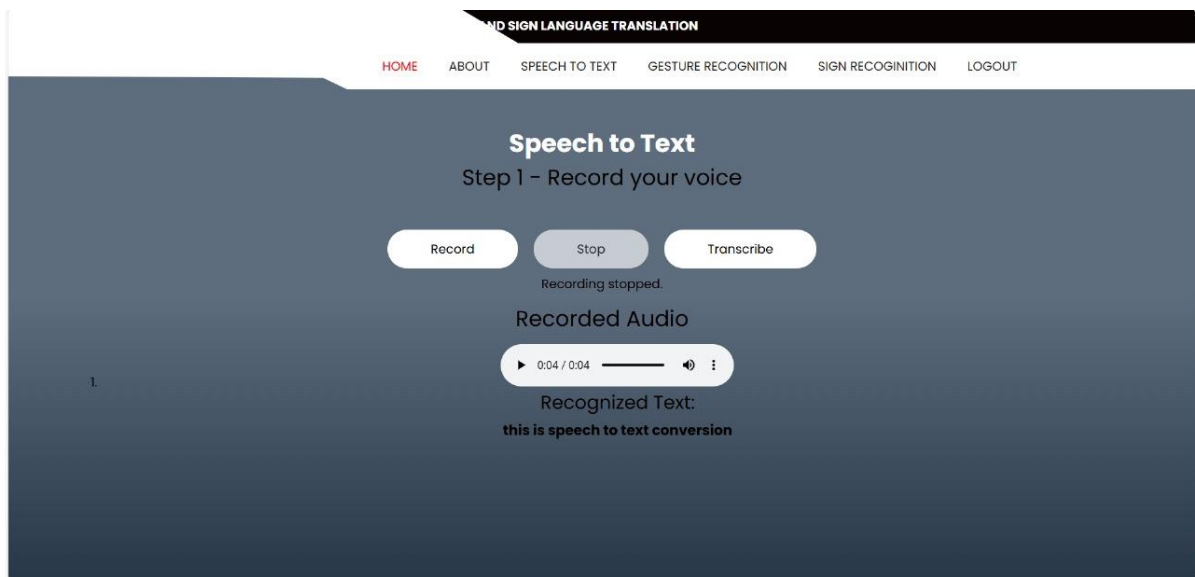
## APPENDIX-B

### SCREENSHOTS

This appendix contains a collection of screenshots that visually document key stages and features of the Sign Language project. Each screen illustrates the system's capability to process inputs, translate sign gestures, and display real-time results. The interfaces are designed to enhance user interaction, accessibility, and overall user experience.

#### Application Home Page

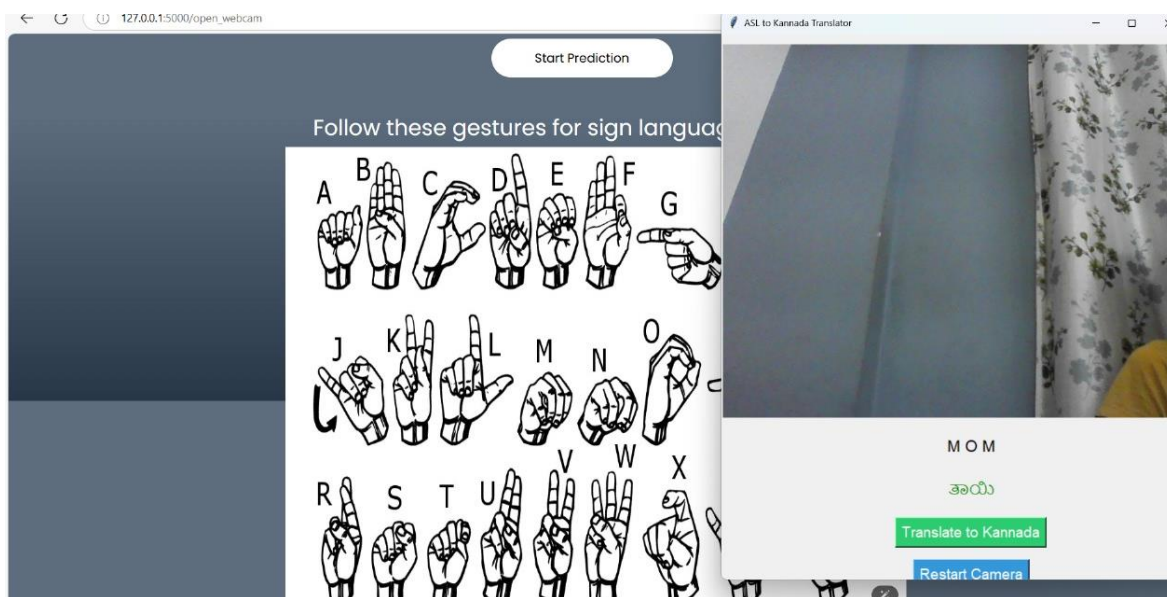
The home page is the main navigation interface of the application. It offers a simple and user-friendly design, allowing users to easily access different features such as sign detection, alphabet reference, and language translation. This centralized layout ensures efficient access to all functionalities.



**Figure B.1: Application Home Page**

#### Kannada Translation from Sign Language

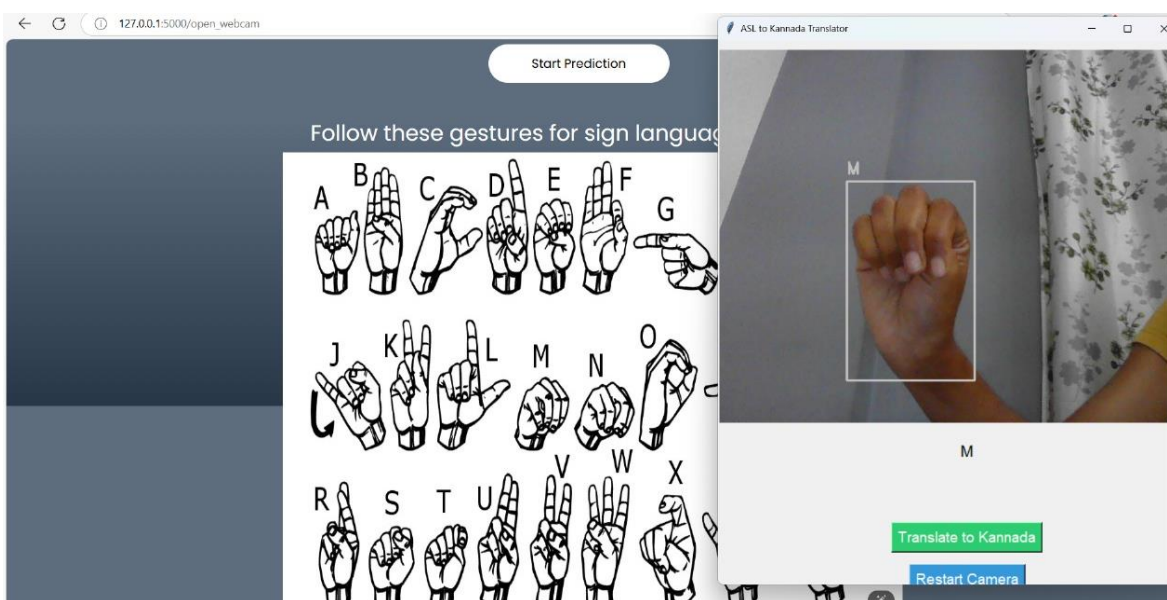
This output illustrates the system's functionality of translating detected sign gestures into Kannada text. The application captures hand gestures using the webcam, processes them through the YOLOv8 model, and displays the recognized output in Kannada, enabling effective communication with Kannada-speaking users.



**Figure B.2: Kannada Translation from Sign Language**

### ASL Alphabet Reference Chart

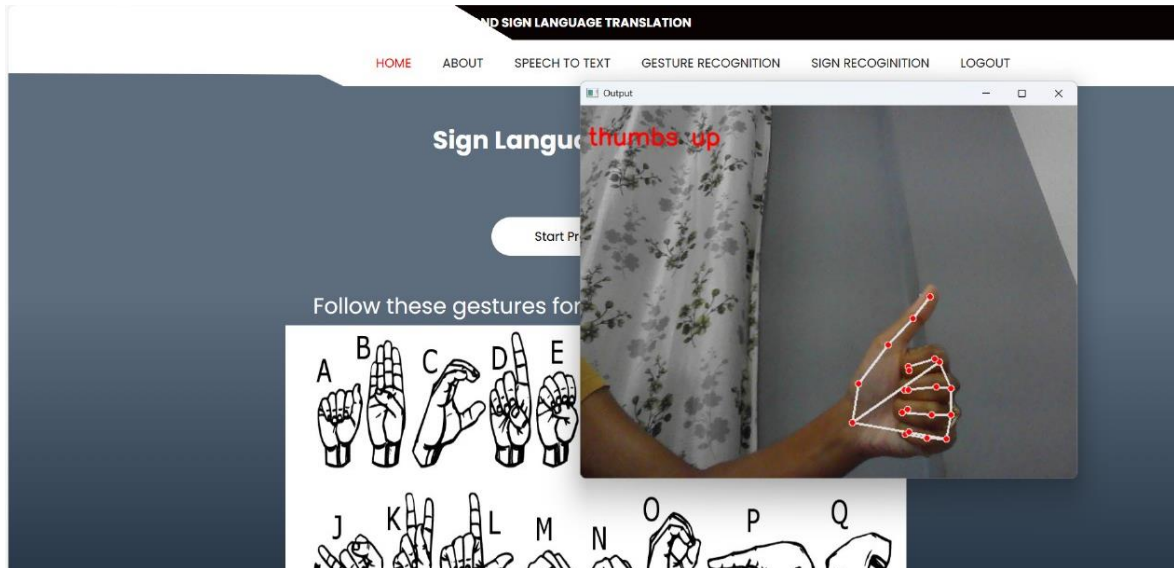
This section provides a visual guide of American Sign Language alphabets, where each letter is represented by a specific hand gesture. This static reference helps users familiarize themselves with the gestures and improves accuracy during live gesture input. It also serves as an educational tool.



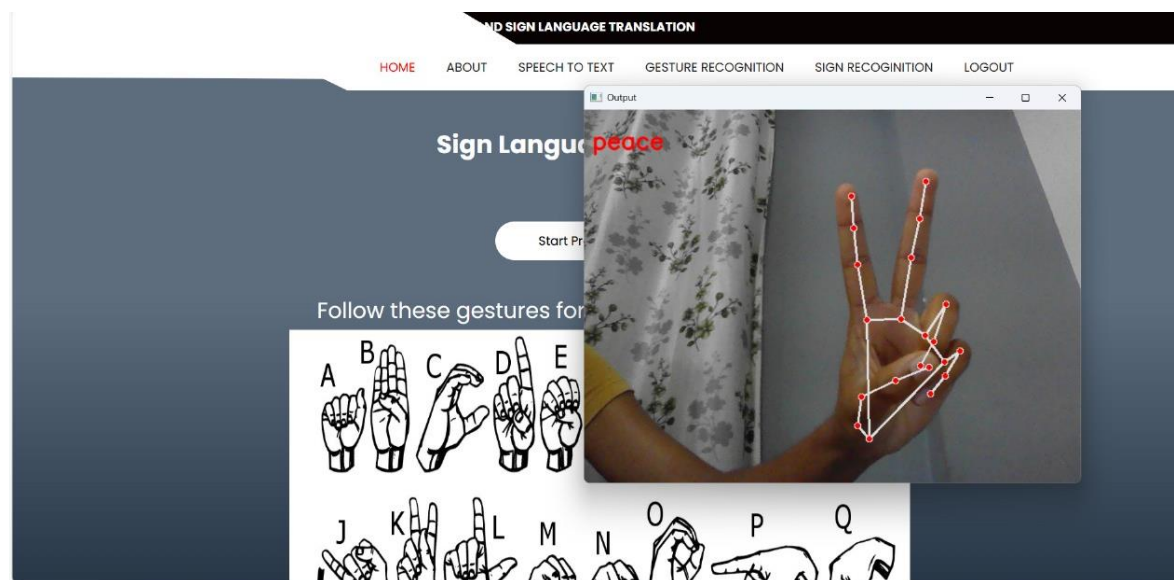
**Figure B.3: ASL Alphabet Reference Chart**

## Real-Time Gesture Detection

This screen captures the live detection process, where the application continuously processes video input to identify and display the corresponding hand gesture in text format. The real-time feedback enhances user interaction and helps validate the model's performance effectively.



**Figure B.4: Real-Time Gesture Detection**



**Figure B.4.1: Real-Time Gesture Detection**

## **APPENDIX-C**

## **ENCLOSURES**

## SDG Mapping

### Project work mapping with Sustainable Development Goals



#### SDG 4: Quality Education

- Promotes inclusive learning, especially for individuals with hearing impairments.
- Supports equal access to quality education through accessible communication technologies.

#### SDG 8: Decent Work and Economic Growth

- Empowers people with communication disabilities using assistive technologies.
- Enhances employment opportunities by improving communication skills and access.

#### SDG 9: Industry, Innovation, and Infrastructure

- Utilizes advanced technologies like YOLOv8, real-time video processing, and multilingual translation.
- Encourages technological innovation in assistive solutions.

#### SDG 10: Reduced Inequalities

- Bridges communication gaps for the hearing-impaired community.
- Promotes equal participation in education, employment, and society.



### **SDG 11: Sustainable Cities and Communities**

- Enhances inclusivity in urban environments by supporting real-time communication for the hearing-impaired.
- Promotes accessibility in public services and community interactions through regional language and sign language integration.

### **SDG 16: Peace, Justice, and Strong Institutions**

- Upholds the rights of people with disabilities through inclusive communication access.
- Ensures equal access to services, justice, and societal engagement.

### **SDG 17: Partnerships for the Goals**

- Encourages collaboration among academic institutions, NGOs, and government agencies focused on accessibility and assistive tech.
- Supports scalable impact through multi-sectoral partnerships that drive sustainable development and inclusion.