Hand Gesture-Based Communication System for Disabled Individual

A PROJECT REPORT

Submitted by

Jeevan Baabu Murugan RA2211026010548
Surya Sathyanarayanan RA2211026010562

Under the Guidance of

Dr. R.Babu

Associate Professor,
Department of Computational Intelligence

in partial fulfillment of the requirementsfor the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in Artificial Intelligence And Machine learning

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2025

Department of Computational Intelligence
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : **B.Tech CSE AIML**

Student Name : **Jeevan Baabu Murugan, Surya Sathyanarayanan**

Registration Number :**RA2211026010548, RA2211026010562**

Title of Work : **Hand Gesture-Based Communication System For Disabled Individual**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is my / our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that we have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

We understand that any false claim for this work will be penalized in accordance with theUniversity policies and regulations.

DECLARATION:
We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is my / our own work, except where indicated by referring, and that we have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGYKATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 21CSP302L - Project report titled **Hand Gesture-Based Communication System for Disabled Individual** is the Bonafide work of **Jeevan Baabu Murugan RA2211026010548, Surya Sathyanarayanan RA2211026010562** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr.R.Babu	**DR. R ANNIE UTHRA**
SUPERVISOR Associate Professor DEPARTMEENT OF COMPUTATIONAL INTELLIGENCE	**PROFESSOR &HEAD** DEPARTMENT OF COMPUTATIONAL INTELLIGENCE

Examiner 1

Examiner 2

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Leenus Jesu Martin M,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson - CS, School of Computing and **Dr. Lakshmi,** Professor and Associate Chairperson -AI, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. R Annie Uthra** , Professor and Head Department Of Computational Intelligence, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, Panel Head, and Panel Members Department of Computational Intelligence, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. Kanipriya M**, Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr.R. Babu** , Department of Computational Intelligence, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff members of CINTEL, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

Authors

ABSTRACT

Communication barriers between hearing-impaired individuals and the rest of society continue to create exclusion and misunderstandings, often limiting access to essential services. Traditional solutions like interpreters or written communication are not always practical during everyday interactions. To address this gap, this project proposes a real-time sign language interpretation system built on affordable, portable hardware — specifically the Raspberry Pi 5. It leverages MediaPipe Hands for accurate hand landmark detection, combined with an LSTM-based deep learning model to recognize both dynamic and static signs with high accuracy. A second model interprets custom gestures that control system actions, such as inserting a word, sending a sentence, or triggering presets like "I need help," removing the need for physical buttons and offering a seamless hands-free interface. Powered by a Flask web server running on the Raspberry Pi, the system allows any mobile device on the same hotspot to access a web app where users can see real-time recognized words, edit sentences manually, customize gesture outputs, and review sentence history with timestamps—all without retraining models or rebooting the device. Designed for real-world usability, the system ensures low latency, high accuracy, offline processing, and energy efficiency, while the wearable form factor (such as an ID card with an integrated camera and microphone) allows discreet and portable use. Beyond technical performance, the system prioritizes user experience and inclusivity by enabling full interaction without keyboards, buttons, or voice commands, making it ideal for environments like hospitals, schools, and public transportation. It empowers individuals to communicate independently, reducing reliance on interpreters or caregivers. Looking ahead, the system's flexible architecture supports future upgrades like multilingual voice output, facial expression analysis, and context-aware AI suggestions, making it a scalable, sustainable, and customizable solution that fosters inclusive communication in diverse personal and professional settings.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

BLE	Bluetooth Low Energy
DFD	Data Flow Diagram
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
SDG	Sustainable Development Goal
SVM	Support Vector Machine
TTS	Text-to-Speech
UI	User Interface
US	User Story
DFD	Data Flow Diagram
TTS	Text-To-Speech
SVOX Pico	Small Voice Engine (TTS engine)
Pi	Raspberry Pi
Flask	Python Web Framework (used implicitly but not abbreviated)

CHAPTER 1

INTRODUCTION

1.1 Introduction to AI-Powered Sign Language Translator

Communication between sign language users and the broader spoken-language population remains a major challenge in everyday life, often resulting in barriers to education, healthcare, and social inclusion. To address this gap, an AI-based communication assistant has been developed, combining machine learning, gesture recognition, and real-time natural language processing into an integrated, wearable system powered by the Raspberry Pi 5. The goal is to provide fluent, real-time translation from sign language to English, allowing deaf and speech-impaired individuals to engage more easily in daily conversations.

At the core of the system is a combination of AI models trained on hand pose data, capable of accurately recognizing both dynamic and static hand gestures. A pose classifier with Long Short-Term Memory (LSTM) networks identifies gesture patterns corresponding to words, while an auxiliary model recognizes control gestures like "add word," "send sentence," and "reset." Once a gesture is recognized, the words are processed through a grammar refinement step using Google Gemini AI, ensuring that the output is grammatically correct and contextually appropriate.

A major strength of the system lies in its real-time interactivity and flexibility. A locally hosted web interface allows users and caregivers to view recognized words, manually edit sentences if needed, modify predefined outputs, and monitor a live log of generated speech outputs. The final output is delivered using text-to-speech synthesis, either through an onboard speaker or via Bluetooth-connected devices, offering adaptability for various environments.

By integrating accurate gesture recognition, smart sentence generation, and versatile voice output, the system offers an accessible, cost-effective, and easy-to-use platform for improving communication among individuals with speech and hearing impairments. It presents a practical solution for use in education, healthcare, public services, and everyday social interactions.

1.2 Motivation

The idea for developing the Hand Gesture-Based Communication System for Disabled Individuals came from observing the real difficulties faced by people with speech and hearing impairments in daily life. In many cases, individuals who rely on hand gestures or sign language find it hard to communicate with others who are unfamiliar with these signs, especially in public places like hospitals, schools, or transport systems.

Even though there are some sign language apps available, most of them are not very practical — they either require internet connectivity, expensive devices, or have limited gesture detection. Many times, they do not provide real-time translation or natural voice output, which makes communication slow and uncomfortable.

Our project specifically aims to solve this gap by creating a portable, real-time gesture recognition system that can instantly convert hand gestures into text and speech. We have used predefined hand gestures and trained our model to recognize them accurately without depending heavily on external servers or costly hardware.

Another key motivation for this project was the realization that not every individual uses the exact same style of gesture. So, we made sure that the system could process variations naturally and still deliver accurate communication. By adding voice output, the system allows users to express themselves clearly and helps them participate more easily in daily conversations.

Overall, our inspiration was to create something simple, affordable, and easy-to-use, especially keeping in mind real-life challenges faced by disabled individuals when they try to express themselves independently.

Apart from helping individuals with disabilities, this project also aims to raise awareness about how technology can bridge communication gaps in society. We focused on using readily available components like a webcam, MediaPipe, and text-to-speech engines to keep the system low-cost and accessible to everyone. By integrating AI-based sentence formation through Gemini, we also made the communication feel more natural and less robotic. Our goal was not just to create a tool, but to develop a supportive system that brings more confidence to users, allowing them to communicate freely without always depending on others. In the future, we also hope to add support for more languages and better personalization options to make the system even more useful across different communities.

1.3 Sustainable Development Goal of the Project

Sustainable Development Goal 4 established by the United Nations emphasizes inclusive and equitable quality education and the promotion of lifelong learning opportunities for all. One of the key enablers of education is communication. For individuals with hearing or speech impairments, the absence of effective communication tools often limits their participation in classrooms, training environments, and collaborative learning spaces. This gap not only affects their academic progress but also impacts their confidence and social inclusion.

To support inclusive education, systems that facilitate real-time interpretation of sign language into speech can play a transformative role. Through the integration of gesture recognition, sentence construction, and voice output technologies, such systems make it possible for non-verbal learners to interact with peers and educators in a seamless and independent manner. This ensures that learners who depend on non-verbal forms of communication can fully participate in academic discussions and classroom interactions, aligning directly with the vision of SDG 4.

Further, when the system includes customizable gesture mappings, editable speech outputs, and visual feedback mechanisms, it becomes adaptable to the unique needs of each user. This feature is especially valuable for educators and caregivers who support students with varying levels of language comprehension and communication abilities. It also enhances the role of technology in enabling differentiated instruction, a core requirement in inclusive educational environments.

Affordability and ease of deployment are also critical factors in scaling inclusive tools. By relying on cost-effective microcomputers like the Raspberry Pi and open-source frameworks, such systems can be deployed in schools, rehabilitation centers, and home-based learning setups—even in resource-constrained regions. This contributes not only to inclusive classroom participation but also supports continuous learning outside of traditional environments, ensuring long-term educational engagement for individuals with disabilities.

Ensuring that every learner has the tools to express themselves is fundamental to creating an inclusive educational environment. When students with speech or hearing impairments are empowered to participate actively through technology-assisted communication, it fosters a sense of belonging and equal opportunity.

1.4 Product Vision Statement

The Wearable Sign Language Translator is primarily designed for individuals with speech or hearing impairments who need an easy-to-use, real-time communication assistant. These users require a compact, hands-free solution that can accurately translate their hand gestures into audible speech without relying on traditional monitors or keyboards. In addition to the primary users, the system also addresses the needs of a secondary audience, including caregivers, teachers, therapists, and inclusive organizations such as hospitals, rehabilitation centers, and schools that support individuals with communication challenges. These stakeholders often seek effective assistive technologies that can bridge the communication gap efficiently.

To meet these requirements, the system focuses on delivering real-time translation from sign language to spoken natural language, ensuring a highly responsive and accurate gesture recognition process. It also offers user-extensible sign-to-word mappings, allowing customization to better fit the personal expressions of different users. Beyond the core functionality, the project addresses secondary needs such as providing a web interface for remote configuration, live editing of preset gestures, and monitoring system activity. Wireless connectivity is also prioritized, with Bluetooth-enabled audio output for seamless integration with headphones or portable speakers. Furthermore, a real-time feedback and logging mechanism is embedded into the system, enabling users and developers to track performance and usage patterns over time.

The core product itself is a wearable, AI-powered sign language translator. It integrates MediaPipe for precise hand tracking, LSTM-based models for robust gesture classification, and Gemini AI for natural sentence construction based on recognized keywords. Additional features enrich the product offering, including Gemini integration for intelligent sentence building, a Flask-based web interface that allows users to manage presets and review usage logs, Bluetooth support for wireless voice output, and a detailed logging system that records all generated sentences along with timestamps. Importantly, users are empowered to edit their preset gestures in real time, providing a personalized and adaptable communication experience.

1.5 Product Goal

The primary aim is to improve communication access for individuals with speech or hearing impairments through a real-time, wearable sign language translation mechanism. The intent is to support smooth and independent interaction in daily life by converting hand gestures into structured, spoken sentences. This removes the reliance on third-party interpreters and provides a direct communication pathway that is accessible, non-intrusive, and easy to use.

Gesture tracking is achieved using MediaPipe for real-time landmark detection, while classification is handled by a deep learning model based on Long Short-Term Memory (LSTM) architecture. Recognized gestures are converted into meaningful sentences with the help of a sentence construction layer, ensuring that output remains grammatically coherent and contextually appropriate. This approach is designed to go beyond basic word- by-word translation, enabling the formation of full sentences for more natural communication.

The system includes features like preset customization for frequently used phrases, gesture history logging, and real-time interface controls. These allow users or caregivers to adapt the behavior of the system according to specific needs. Voice output is delivered through both built-in audio and Bluetooth-supported devices, allowing flexible use in public and private scenarios alike.

Affordability and scalability remain central to the design approach, making it suitable for deployment in schools, healthcare centers, and community spaces. The focus is on promoting accessibility, autonomy, and inclusion, especially for those who may not have access to commercial assistive technologies. In doing so, the work contributes meaningfully toward inclusive education and reduced inequalities in communication.

Ultimately, the product's goal is to create a low-cost, scalable communication aid that promotes inclusion, independence, and dignity. It transforms sign language from a visually bound system into one that can speak, helping bridge communication gaps and fostering better understanding in communities.

1.6 Product Backlog

The following table 1.1 represents the user stories

Table 1.1 Product backlog table

S.No	User Stories
#US 1	As a hearing-impaired user, I want the device to recognize my sign language gestures so that I can communicate easily.
#US 2	As a caregiver, I want to receive text translations of sign language so that I can better understand and assist.
#US 3	As a developer, I want to train the AI model to support multiple languages so that it can be used globally.
#US 4	As a user, I want the device to provide an audio output of the recognized signs so that non-signers can understand.
#US 5	As a researcher, I want the system to log gesture data so that it can be analyzed for improvement.
#US 6	As a teacher, I want to customize and add new gestures to the system for specific educational needs.
#US 7	As a user, I want to create custom gesture-based shortcuts to perform frequent actions quickly.
#US 8	As a user, I want to see the battery percentage on the display so that I can monitor device usage.
#US 9	As a user, I want to receive low battery alerts so that I can charge the device on time.
#US 10	As a user, I want to enable or disable the background noise filter for better audio output clarity.

The product backlog for the AI-Based Social Media Communication Application was created using Agile principles and is visually represented in Figure 1.1. It outlines all the key user stories that define the system's intended features and functionality.

Each user story is crafted from a specific user perspective and includes:

- MoSCoW Prioritization to highlight the importance of each feature (Must Have, Should Have, etc.)
- Functional Requirements such as gesture recognition, audio/text output, and custom gesture creation
- Non-Functional Requirements covering aspects like response time, battery efficiency, and multilingual support
- Acceptance Criteria that clearly define how the feature will be validated
- Effort Estimation and Status to assist in planning, tracking, and sprint execution

This structured backlog ensures that development is focused on user-centric, accessible solutions, particularly benefiting individuals with hearing and speech impairments through smart AI-powered communication tools.

Figure 1.1 represents planner board that has our functional implementation

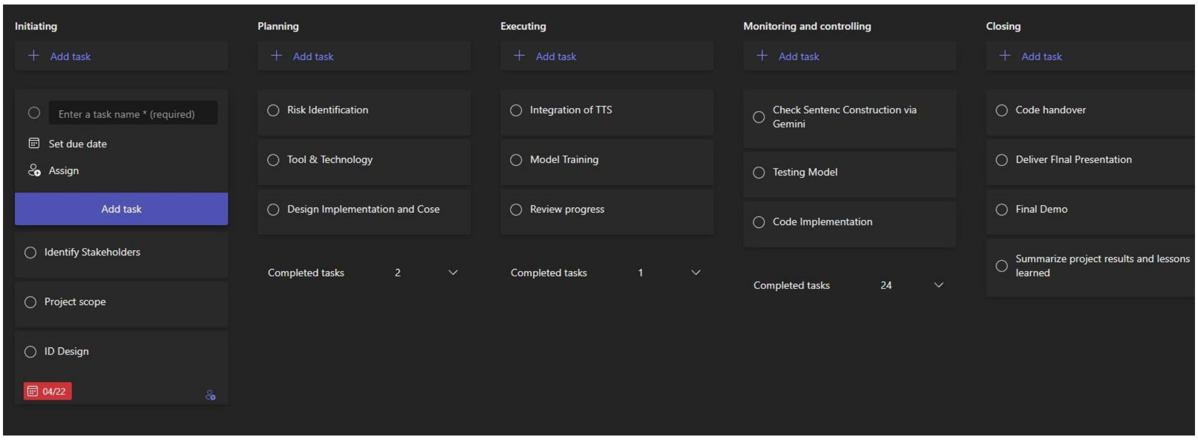


Figure 1.1 MS Planner Board of Hand gesture Recognition

1.7 Product Release Plan

The following Figure 1.2 depicts the release plan of the project

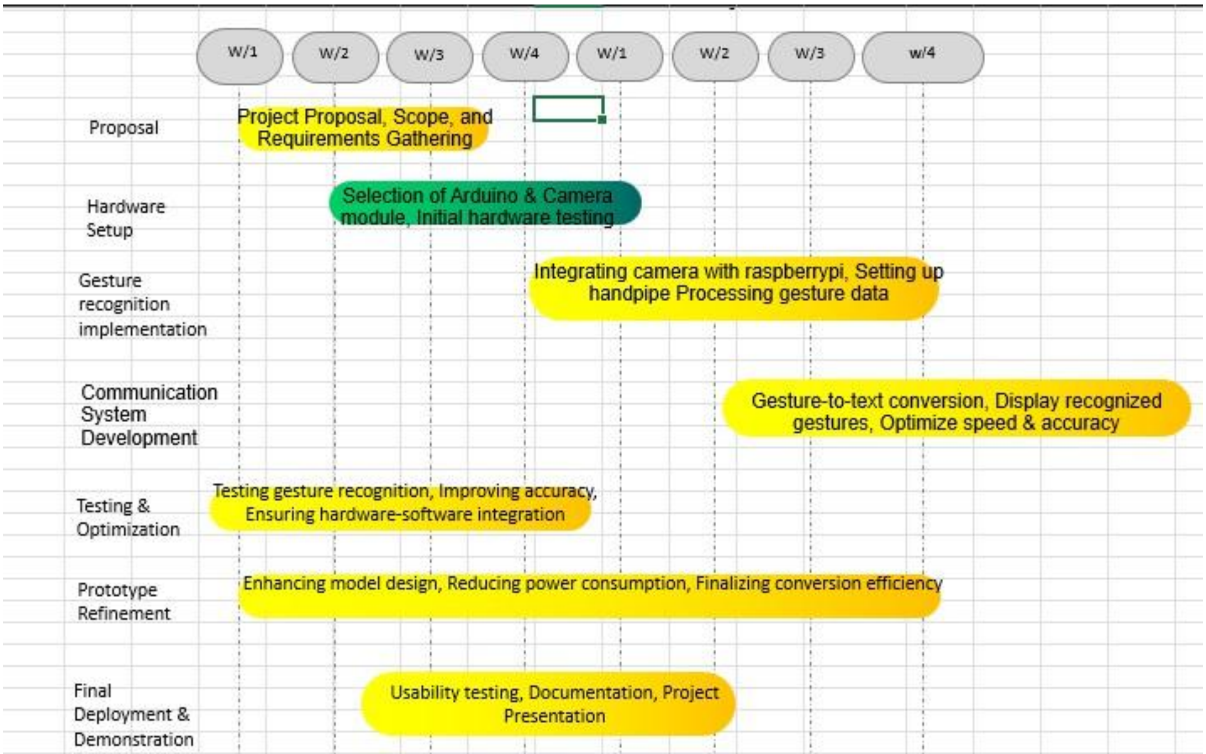


Figure 1.2 Release plan of Hand Gesture Recognition

CHAPTER 2

SPRINT PLANNING AND EXECUTION

2.1 Sprint 1

2.1.1 Sprint Goal with User Stories of Sprint 1

The goal of Sprint 1 is to set up the project environment and implement the basic gesture recognition module using a pre-trained model.

The following table 2.1 represents the detailed user stories of the sprint 1

Table 2.1 – Detailed User Stories of Sprint 1

S. No	Detailed User Stories
US #1	As a developer, I want to set up the Raspberry Pi environment with camera integration so that I can capture hand gestures in real time.
US #2	As a user, I want the system to recognize predefined hand gestures so that I can start basic communication.
US #3	As a developer, I want to process hand landmarks using MediaPipe so that I can feed meaningful data to the gesture model.

Planner Board representation of user stories are mentioned below figures 2.1

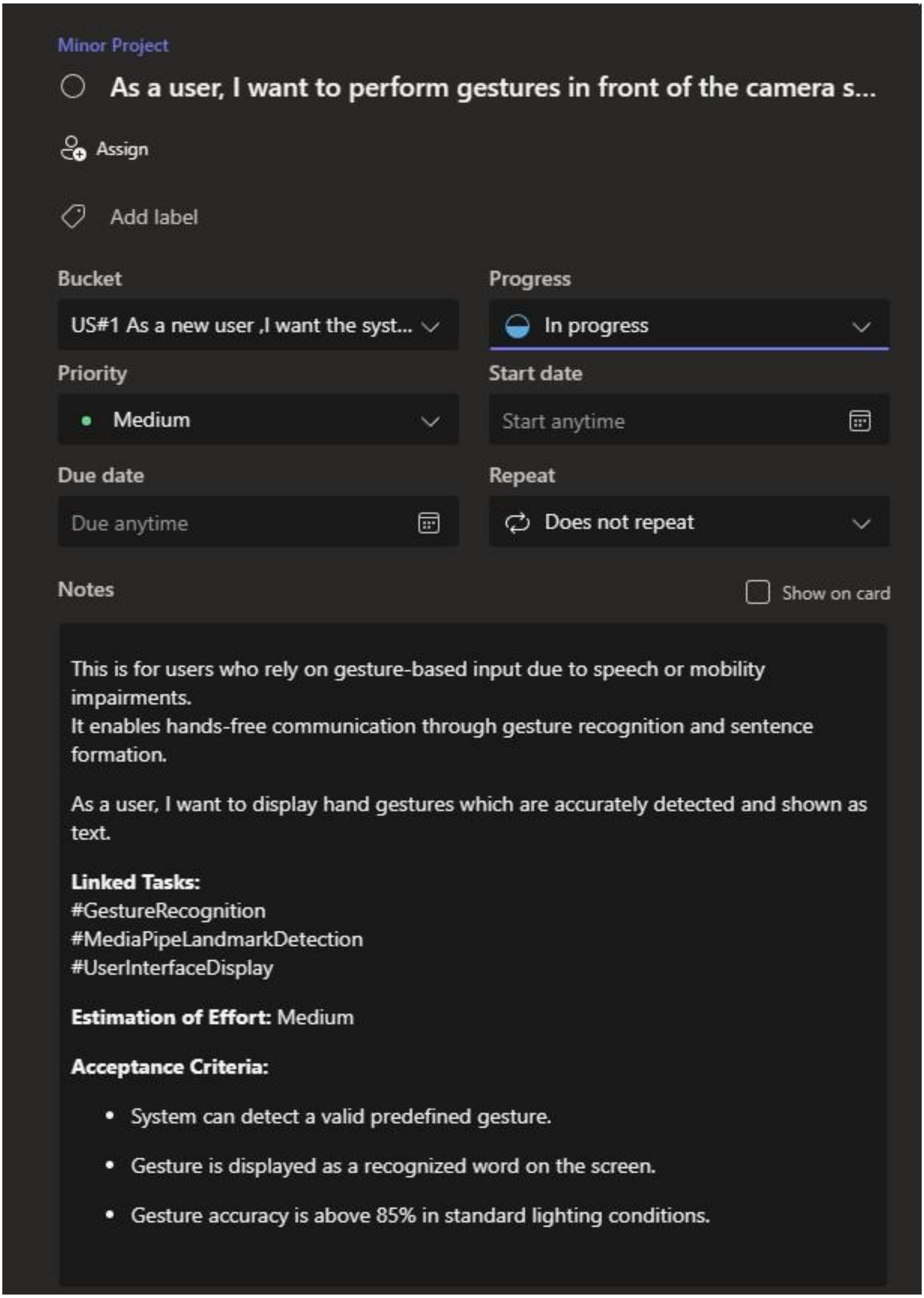


Figure 2.1 user story for Hand gesture performance in camera

2.1.2 Functional Document

Introduction

First sprint centers on building the technical foundation required for real-time gesture recognition in the Wearable Sign Language Translator. Key tasks involve setting up the Raspberry Pi environment, integrating the camera for live gesture capture, processing hand landmarks using the MediaPipe framework, and preparing data pipelines for the gesture classification model. These tasks lay the groundwork for achieving reliable and efficient hand gesture detection in future iterations.

User Stories and Functional Goals

Functionality: The Raspberry Pi 5 will be configured with necessary system packages, camera drivers, and libraries to ensure stable video stream acquisition. Testing will include capturing high-frame-rate video feeds under varied lighting conditions to ensure the hand gestures are consistently detected by the onboard camera module.

Functionality: A limited set of gestures—mapped to essential phrases—will be predefined. The system will match real-time hand pose data to these gestures and output the corresponding text. Accuracy testing will be performed across multiple users to ensure consistency and reduce false positives.

Functionality: MediaPipe Hands will be used to detect and track 21 key hand landmarks. These points will be preprocessed into a normalized, structured input format suitable for the LSTM model. Data cleaning techniques such as landmark smoothing and noise filtering will be applied to improve model input quality.

Deployment and Validation

Initial deployments will be tested using controlled indoor environments. Data collection from the Raspberry Pi camera will be validated by manually comparing detected landmarks with expected hand poses. Early TTS (Text-to-Speech) integration will also be checked to ensure that recognized gestures trigger audible outputs, forming a basic, functional communication loop.

Authorization Matrix

The following table 2.2 represents the Authorization Matrix, User Level

Table 2.2 Authorization Matrix

ROLE	ACCESS LEVELS
END USER	Use of the wearable for gesture input and audio output
MOBILE PHONE USERS	View gesture history, configure gesture mappings
ADMIN	Add/edit gesture predefined profiles, manage system updates.

1. Assumptions

- Raspberry Pi 5 and battery module will be available and functional throughout the sprint.
- The MediaPipe and LSTM models will perform well under normal lighting conditions.
- Team members are experienced with Python, NLP, TensorFlow, and embedded systems.
- The mobile app and Pi will communicate over Bluetooth/Wi-Fi without significant latency.

2.1.3 Architecture Document

2. Chosen Architecture: Microservices Architecture

2.1 Justification for Microservices

The decision to implement a Microservices Architecture for the Hand Gesture Recognition system stems from several key technical and strategic factors. A major advantage is scalability—each core function, including gesture recognition, text processing, speech synthesis, and mobile app communication, is designed as an independent service. This allows different components to scale individually based on their specific load and performance needs. For example, if gesture recognition requires more computational resources than speech output, only that service needs to scale, without impacting the others. This modular setup not only improves performance but also optimizes resource usage. In addition to scalability, flexibility is a critical benefit. Since services are loosely coupled, individual parts of the system, such as the gesture detection module or battery monitoring system, can be developed, modified, or upgraded independently without risking downtime or cascading failures across the entire system.

Another core reason for adopting this architecture is the freedom of technology choice for each service. Instead of locking into a single stack, the system leverages specialized tools where appropriate—such as TensorFlow Lite for efficient machine learning inference and simple JSON files for lightweight, fast, and easy-to-manage local data storage. For device communication, technologies like Bluetooth (BLE/A2DP) are utilized to maintain seamless and reliable connectivity. This diverse toolset ensures that each service is optimized for its role, enhancing the overall robustness and adaptability of the platform. Furthermore, the microservices model promotes parallel development across different teams. Work on gesture recognition, mobile applications, speech output, and battery monitoring can proceed independently, dramatically accelerating the development process. Ultimately, microservices architecture empowers the system to be more scalable, flexible, and future-ready, making it well-suited for continuous improvement and expansion over time.

3. Detailed Diagrams

3.1 Use Case Diagram

The figure 2.2 represents Use case Diagram

User performs sign language → Triggers the system.

Capture Gesture → Detects hand movement.

Process Gesture → Converts the gesture into text.

Convert to Speech → Converts the text into speech.

Figure 2.2 represents the use case diagram

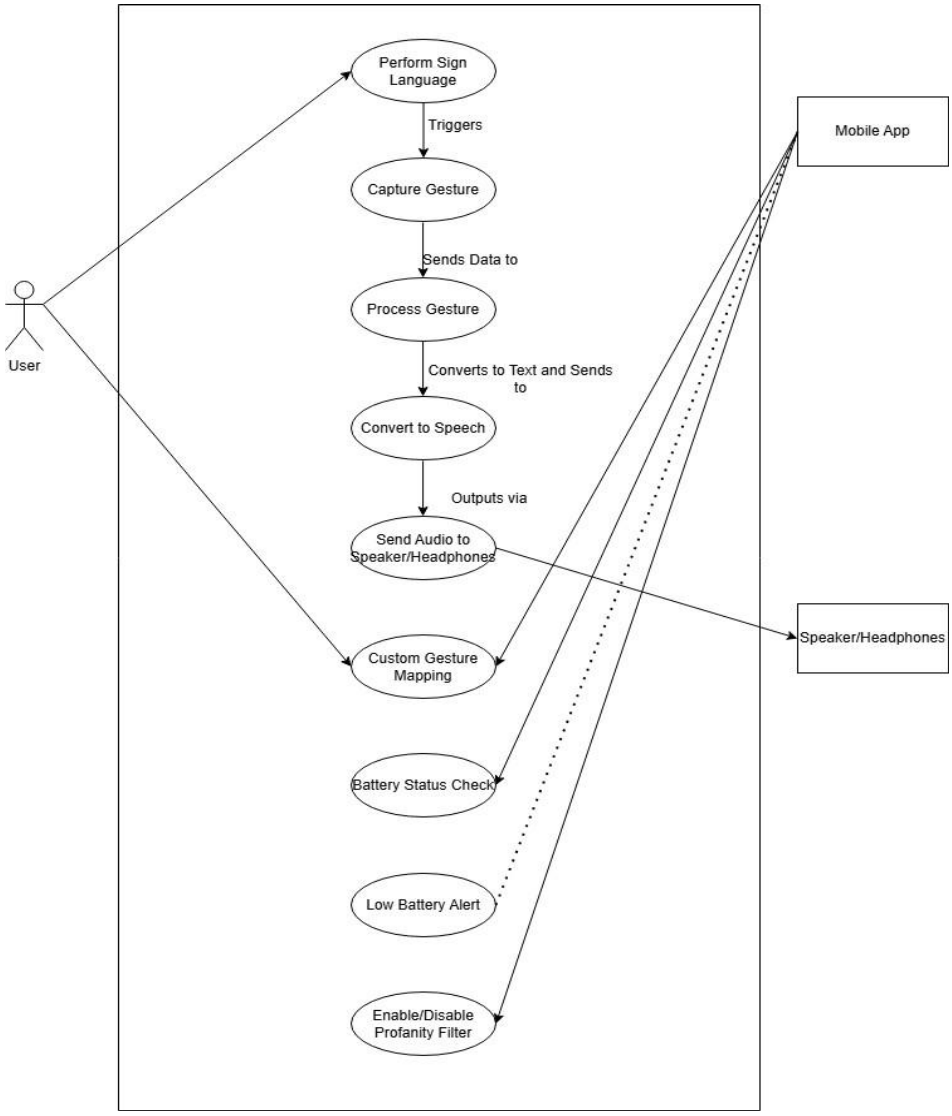


Figure 2.2 Use Case Diagram

2.1.4 Functional Test Cases

The following table 2.3 represents the detailed functional Test Case of the sprint 1

Table 2.3 Detailed Functional Test Case

Feature	Test Case	Steps to execute test case	Expected Output	Actual Output	Status	More Information
Gesture Recognition	Recognize Predefined Gesture	1. Start application 2. Show a trained gesture in front of the webcam	Gesture is recognized	Recognized correctly	Pass	Using trained model (pose_classifier_final.h5)
Gesture Recognition	Detect Unknown Gesture	1. Start application 2. Show a random or untrained hand gesture	Displays message or shows no confidence in prediction	No gesture or low confidence shown	Pass	System ignores unknown gestures
Gesture to Sentence	Add Word to Sentence	1. Show gesture 2. Press SPACE key to add it to the sentence	Word added to sentence list	Word added and confirmed in terminal	Pass	One word added per gesture per cooldown
Voice Output	Convert Sentence to Speech	1. Press 'S' after forming sentence 2. Wait for Gemini to process and convert to sentence 3. Listen to the voice output	AI-generated sentence is spoken aloud using text-to-speech	Gemini responds and audio played	Pass	Uses Gemini + pyttsx3 for speech
System Stability	Cooldown on Gesture Recognition	1. Repeat same gesture rapidly 2. Observe response	Recognizes only once during cooldown period	System prevents multiple fast repeats	Pass	Uses cooldown frame logic
System Startup	Webcam and Model Initialization	1. Run the script 2. Check if webcam opens and model loads	Webcam starts, model loads successfully	Webcam and window appear	Pass	Tested under normal lighting
Keyboard Events	Trigger Speak/Reset with Key	1. Press 'S' to speak 2. Press 'Q' to quit	Gemini responds with a complete sentence and audio App quits on 'Q'	Works as expected	Pass	Keyboard-driven interaction
Text-to-Speech Engine	Clarity of Voice Output	1. Press 'S' after forming a sentence	System speaks the sentence clearly	Audio is understandable	Pass	Uses pyttsx3, tested on speakers and headphones
API Integration	Gemini API Usage for	1. Form a sentence using gestures 2. Press 'S' 3. Wait for Gemini to generate a	Converts keywords to a grammatically correct sentence	Returns meaningful sentence	Pass	Gemini API key loaded using dotenv

2.1.5 Committed Vs Completed User Stories

The following figure 2.3 represents the committed Vs Completed User Stories of sprint 1

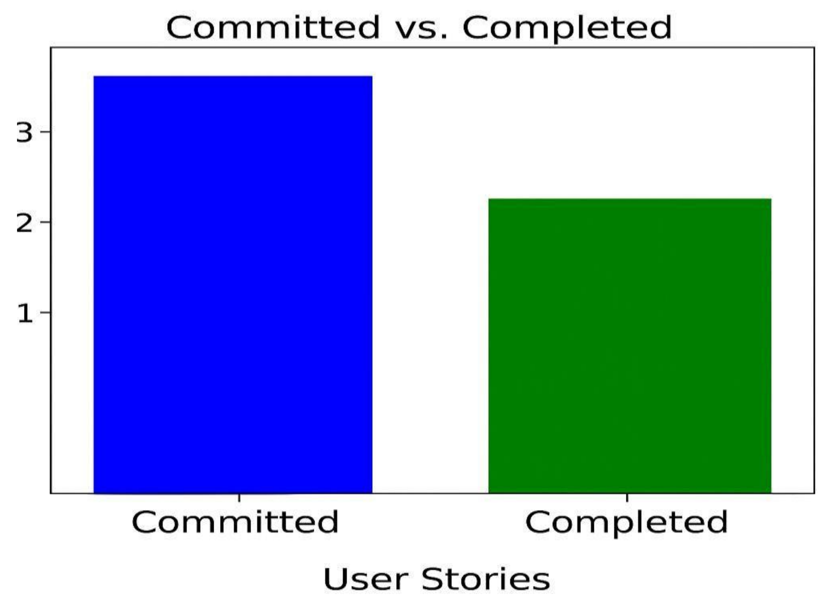


Figure 2.3 Bar graph for Committed Vs Completed User Stories

2.1.6 Sprint Retrospective

Sprint 1 focused on establishing the foundational environment for the project and implementing a basic gesture recognition module using a pre-trained model. The team began by finalizing the hardware components, including the Raspberry Pi 5, Camera Module 3, and a reliable power bank setup for portable use. The software environment was successfully configured with Python, OpenCV, TensorFlow Lite, and MediaPipe. The MediaPipe Hands module was deployed for real-time landmark tracking, which accurately detected hand movements and gestures under stable lighting. Basic gesture detection was tested and validated, forming the backbone of the recognition system. Version control using Git was also set up early, which helped manage progress and code iterations effectively throughout the sprint.

However, a few challenges emerged. The recognition accuracy dropped in low-light conditions, and frame rate inconsistencies were noticed when running simultaneous processes. Additionally, early testing lacked a standardized naming scheme for gestures, which briefly hindered clarity. Despite these hurdles, the sprint was a valuable learning experience. The team realized the importance of modular testing and the limitations of pre-trained models in handling complex gestures. Moving forward, the plan is to expand the system with gesture-to-text mapping, begin integrating Gemini AI for sentence construction, and prepare for BLE-based mobile app interaction. Sprint 1 concluded with a stable environment and a working prototype, successfully meeting its intended objectives.

It became evident that gesture recognition is highly sensitive to environmental factors such as lighting and hand positioning, emphasizing the need for adaptable solutions in future stages. Testing each module separately—gesture recognition, audio output, UI—helped isolate bugs and made debugging more manageable. Additionally, while the pre-trained MediaPipe model proved effective for basic gestures, it lacked the flexibility to handle complex or dynamic movements, indicating the need for custom training in later sprints.

Moving into the next sprint, the team plans to introduce gesture-to-text mapping, begin integrating Gemini AI for sentence construction, and prepare the structure for user-defined gesture presets. Plans for Bluetooth communication with a mobile app interface will also be initiated. Overall, Sprint 1 was a productive phase that successfully established the core environment and validated critical components, ensuring that the project is on track to build a scalable and accessible communication system for users with hearing or speech disabilities.

2.2 SPRINT 2

2.2.1 Sprint Goal with User Stories of Sprint 2

The goal of Sprint 2 is to build the interface for forming gesture-based sentences and enable speech output using text-to-speech conversion.

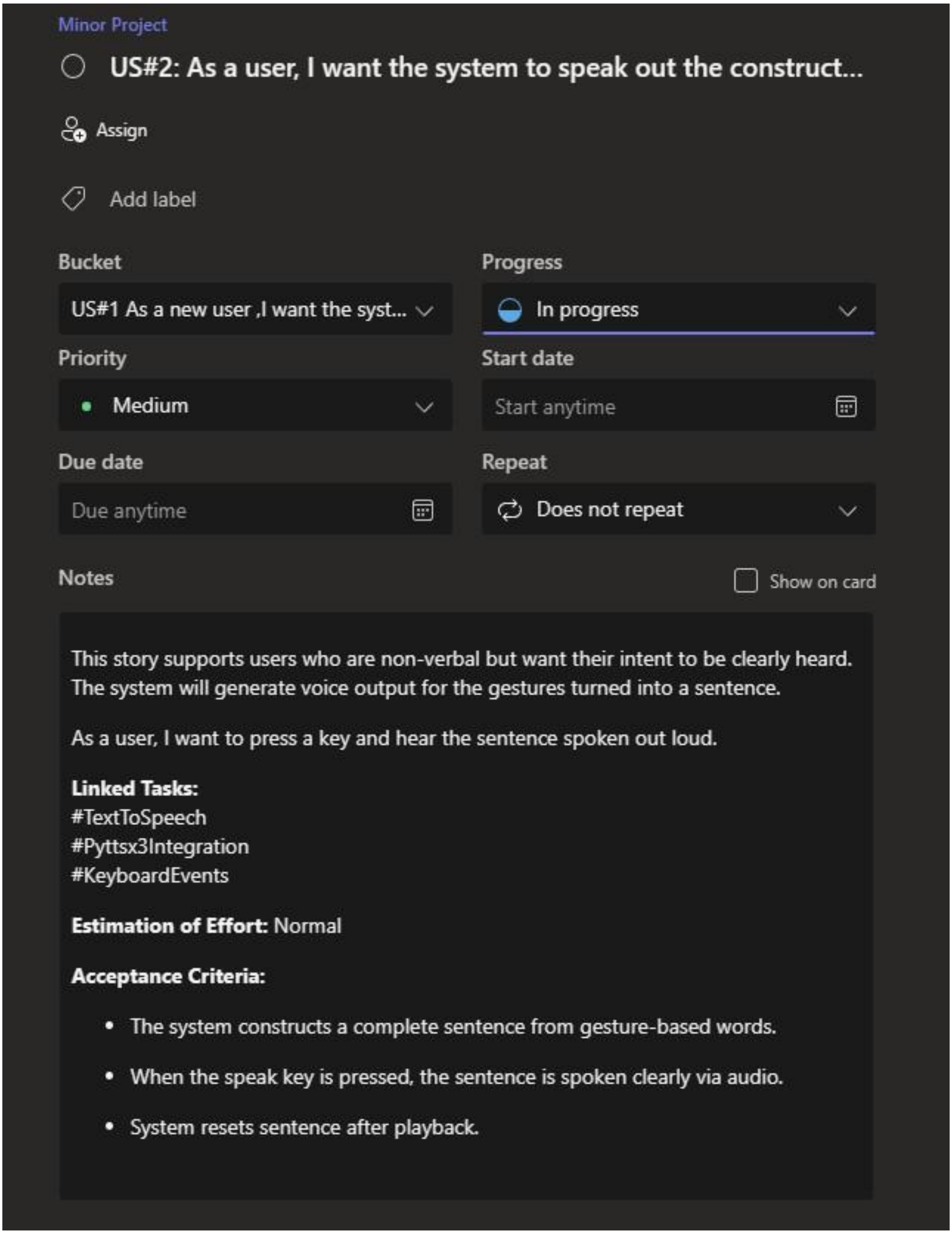
The following table 2.4 represents the detailed user stories of the sprint 2

Table 2.4 – Detailed User Stories of Sprint 2

S. No	Detailed User Stories
US #4	As a user, I want to form sentences using gestures and a keyboard shortcut so that I can construct meaningful phrases.
US #5	As a user, I want the system to speak the constructed sentence using voice output so that I can communicate verbally.
US #6	As a user, I want to clearly view the detected gesture and built sentence on screen so that I can track my input easily.

Planner Board representation of user stories are mentioned below figure 2.4

Figure 2.4 user story for sentence conversion in camera



2.2.2 Functional Document

Gesture-to-Speech Conversion Description

Once a gesture is recognized and classified into a word or phrase, it is converted into speech output using a Text-to-Speech (TTS) engine (e.g., eSpeak). The system ensures real-time response with minimal latency.

User Story

As a speech-impaired user, I want my gestures to be converted into spoken words so that I can communicate with people who do not understand sign language.

2.2.3 Architecture Document

- Raspberry Pi 5 acts as the central processing unit, connecting all components.
- Machine Learning Model (TensorFlow Lite,LSTM & MediaPipe) processes hand gestures. Ultra-Wide Camera Module captures hand movements.
- Audio Output Module (Speaker & Bluetooth A2DP) plays generated speech.
- Bluetooth Module 1 (BLE for Mobile App Communication) connects with the mobile app. Bluetooth Module 2 (A2DP for Audio Output to Headphones/Speaker) transmits audio.
- Mobile App handles custom gestures, battery status, and profanity filtering. SQLite Database stores gesture mappings and logs.
- Battery Module measures battery levels, sends alerts, and supports Bluetooth BLE. Speaker/Headset plays the generated speech via Bluetooth audio.

Figure 2.5 depicts the system architecture integrates gesture recognition, real-time processing, local storage, and wireless audio output on a Raspberry Pi 5 to enable seamless communication for speech- and hearing-impaired users.

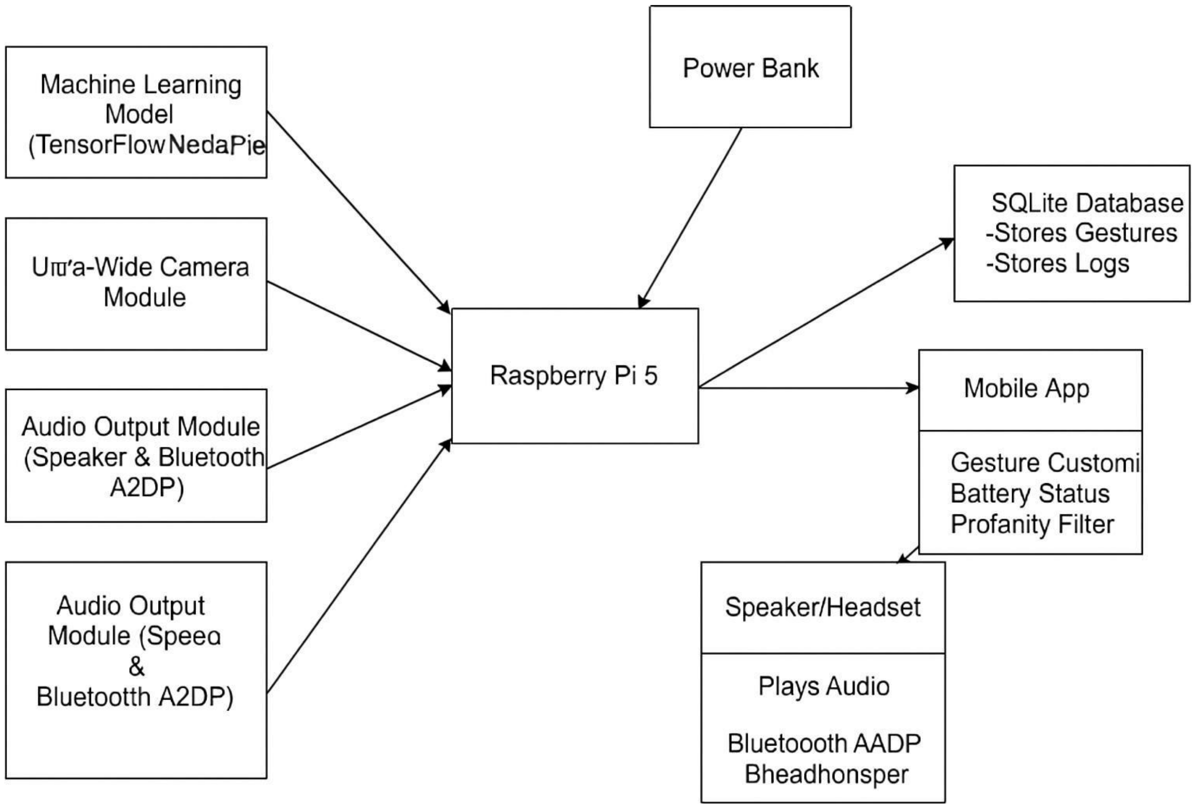


Figure 2.5 Component Diagram

2.3 Deployment Diagram

2.3.1. Central Processing Unit: Raspberry Pi 5

- Acts as the main processing unit.
- Receives input from the Raspberry Pi Camera Module.
- Runs the machine learning model for gesture recognition.
- Connects to external devices via Bluetooth modules.
- Stores and retrieves data from the SQLite database.

2.3.2. Input Component: Raspberry Pi Camera Module 3

- Captures real-time images of hand gestures.
- Sends captured images to the Raspberry Pi for processing.

2.3.3. Machine Learning Model (Gesture Recognition & Classification)

- Uses MediaPipe Hand Tracking to extract 21 hand landmarks.
- Converts hand landmarks into structured data for classification.
- Trains a classifier (Random Forest, SVM, or LSTM) for recognizing gestures.
- Runs TensorFlow Lite for optimized on-device inference.

2.3.4. Text-to-Speech (TTS) Module

- Converts recognized text into speech.
- Uses PicoTTS (SVOX Pico) for speech synthesis.
- Sends audio output via Bluetooth to a speaker or headset.

2.3.5. Gesture & Settings Storage

- Uses an SQLite database to store gestures, user preferences, and settings.
- Ensures gestures are retrievable for real-time conversion.

2.3.6. Mobile App

- Provides an interface for users to customize gestures.
- Displays battery status of the wearable device.
- Implements a profanity filter for speech output.

2.3.7. Speaker/Headset (Bluetooth Audio A2DP)

- Receives the text-to-speech audio output.
- Plays the translated speech for the user.

2.3.8. Battery Management System

- Uses a Power bank to actively run the system
- Sends battery percentage and alerts to the Raspberry Pi.
- Communicates with the mobile app for battery monitoring.

Figure 2.6 The Raspberry Pi 5-based system captures hand gestures using a camera, processes them with machine learning, and outputs speech via Bluetooth or mobile app integration for real-time communication support

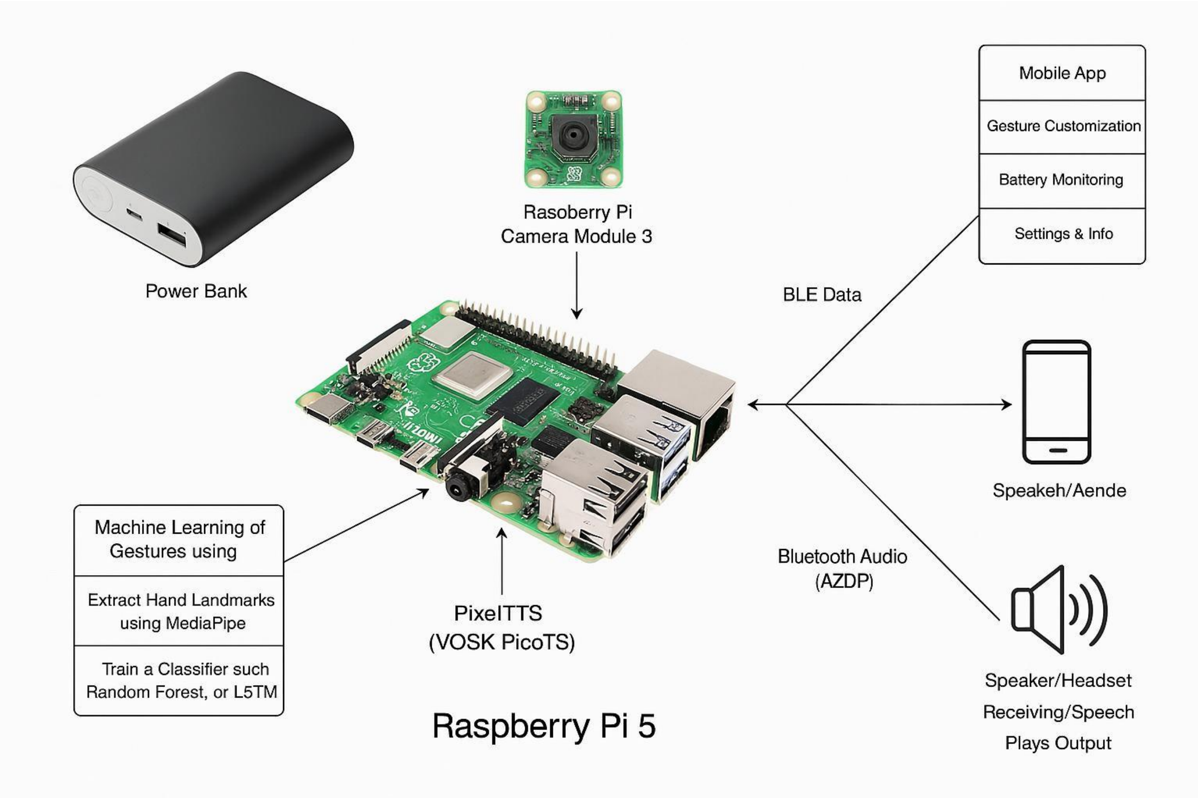


Figure 2.6 Architecture Diagram

2.2.4 Committed vs Completed User Stories

In Figure 2.7 The chart compares the number of user stories committed versus those successfully completed during the sprint execution

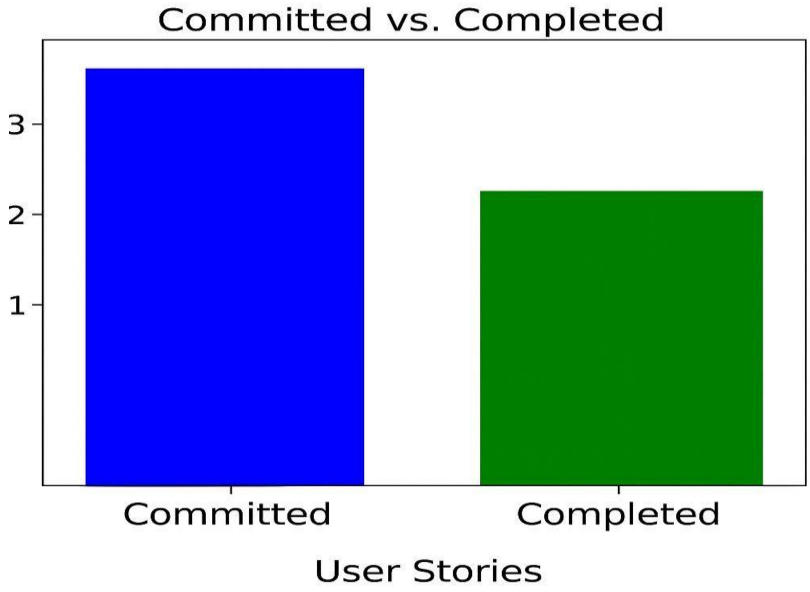


Figure 2.7 Bar graph for Committed Vs Completed User Stories

2.2.5 Sprint Retrospective

Sprint 2 focused on elevating the prototype from simple gesture recognition to a functional communication system capable of forming sentences and delivering speech output. The primary goal was to design a user interface that could process a sequence of recognized gestures and display them as editable text. The team developed a Flask-based web interface, which provided real-time updates of detected gestures and allowed users or caregivers to review and modify the sentence being formed. This interface marked a significant step toward making the system user-friendly and interactive, ensuring that recognized gestures could be assembled into coherent and meaningful expressions.

In parallel, the team worked on integrating Text-to-Speech (TTS) capabilities into the system using the lightweight PicoTTS engine. Once a complete sentence was formed through the interface, it was passed to the TTS module to generate audible speech. This speech output could be heard via the Raspberry Pi's audio jack or streamed wirelessly using Bluetooth A2DP to compatible speakers or headsets. This eliminated the need for external display or manual interpretation, enabling real-time verbal communication directly from the device, thus enhancing its portability and practicality in real-world scenarios

CHAPTER 3

RESULTS AND DISCUSSION

3.1 Project Outcomes

Figure 3.1 contains the UI Web Interface to detect words, construct sentences, send them for ai processing and configure custom preset mappings

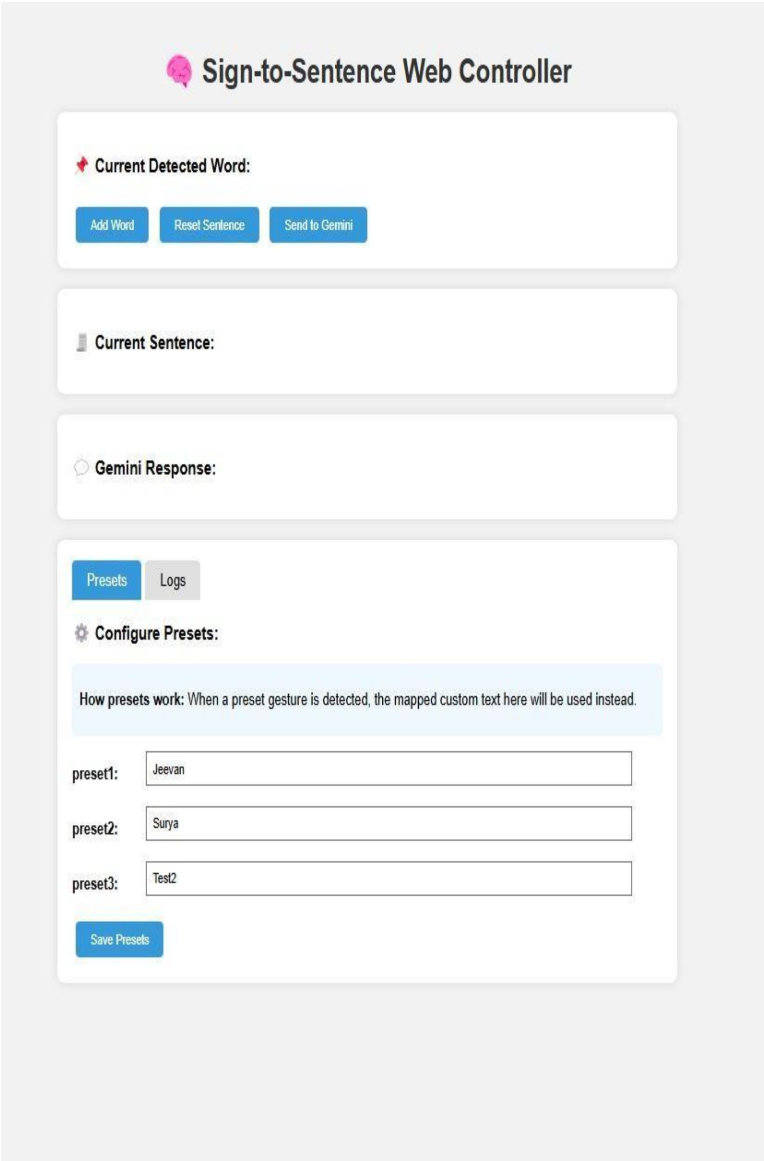
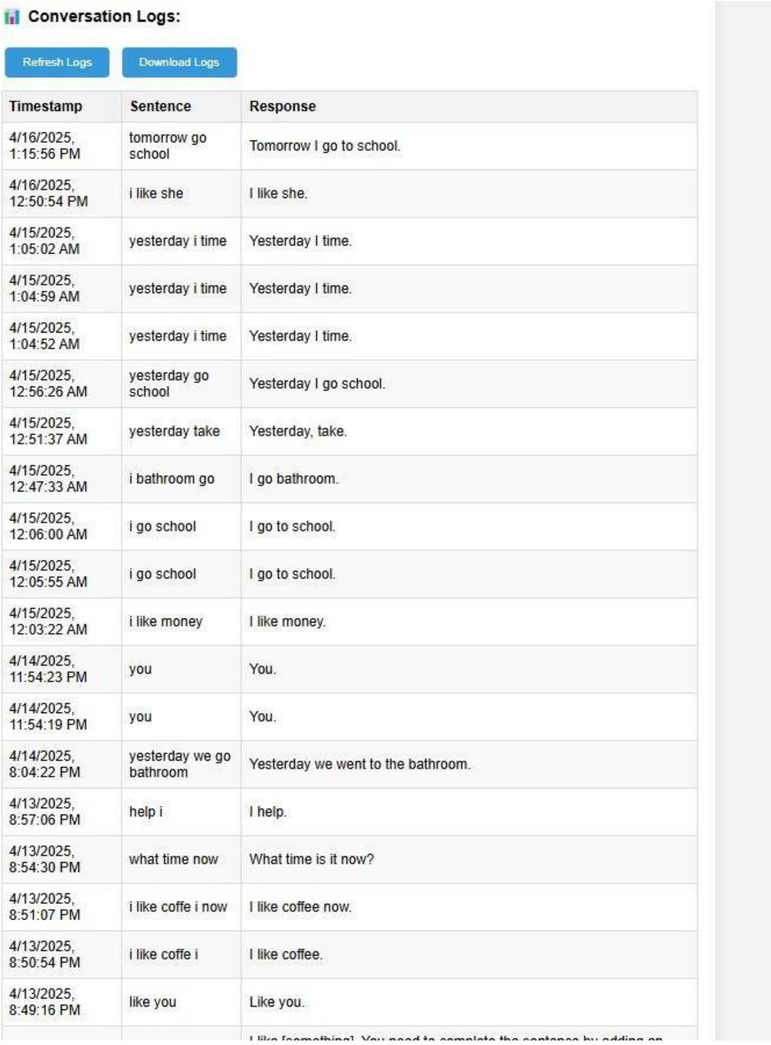


Figure 3.1 UI Front end

Figure 3.2 shows the conversation logs in the UI web application and track the data



📊 Conversation Logs:

Refresh Logs | Download Logs

Timestamp	Sentence	Response
4/16/2025, 1:15:56 PM	tomorrow go school	Tomorrow I go to school.
4/16/2025, 12:50:54 PM	i like she	I like she.
4/15/2025, 1:05:02 AM	yesterday i time	Yesterday I time.
4/15/2025, 1:04:59 AM	yesterday i time	Yesterday I time.
4/15/2025, 1:04:52 AM	yesterday i time	Yesterday I time.
4/15/2025, 12:56:26 AM	yesterday go school	Yesterday I go school.
4/15/2025, 12:51:37 AM	yesterday take	Yesterday, take.
4/15/2025, 12:47:33 AM	i bathroom go	I go bathroom.
4/15/2025, 12:06:00 AM	i go school	I go to school.
4/15/2025, 12:05:55 AM	i go school	I go to school.
4/15/2025, 12:03:22 AM	i like money	I like money.
4/14/2025, 11:54:23 PM	you	You.
4/14/2025, 11:54:19 PM	you	You.
4/14/2025, 8:04:22 PM	yesterday we go bathroom	Yesterday we went to the bathroom.
4/13/2025, 8:57:06 PM	help i	I help.
4/13/2025, 8:54:30 PM	what time now	What time is it now?
4/13/2025, 8:51:07 PM	i like coffe i now	I like coffee now.
4/13/2025, 8:50:54 PM	i like coffe i	I like coffee.
4/13/2025, 8:49:16 PM	like you	Like you.

Figure 3.2 Conversation Log Page

Figure 3.3 shows the hand recognition in the camera module

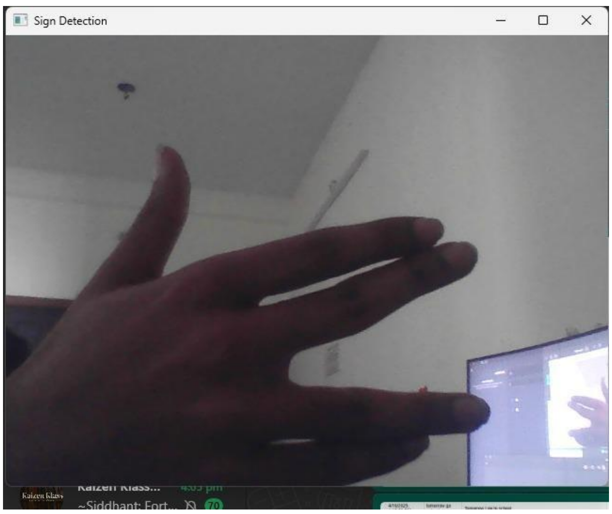


Figure 3.3 Hand gesture Recognition

3.2 Committed Vs Completed User stories

Figure 3.4 contains the bar graph of committed vs completed user stories

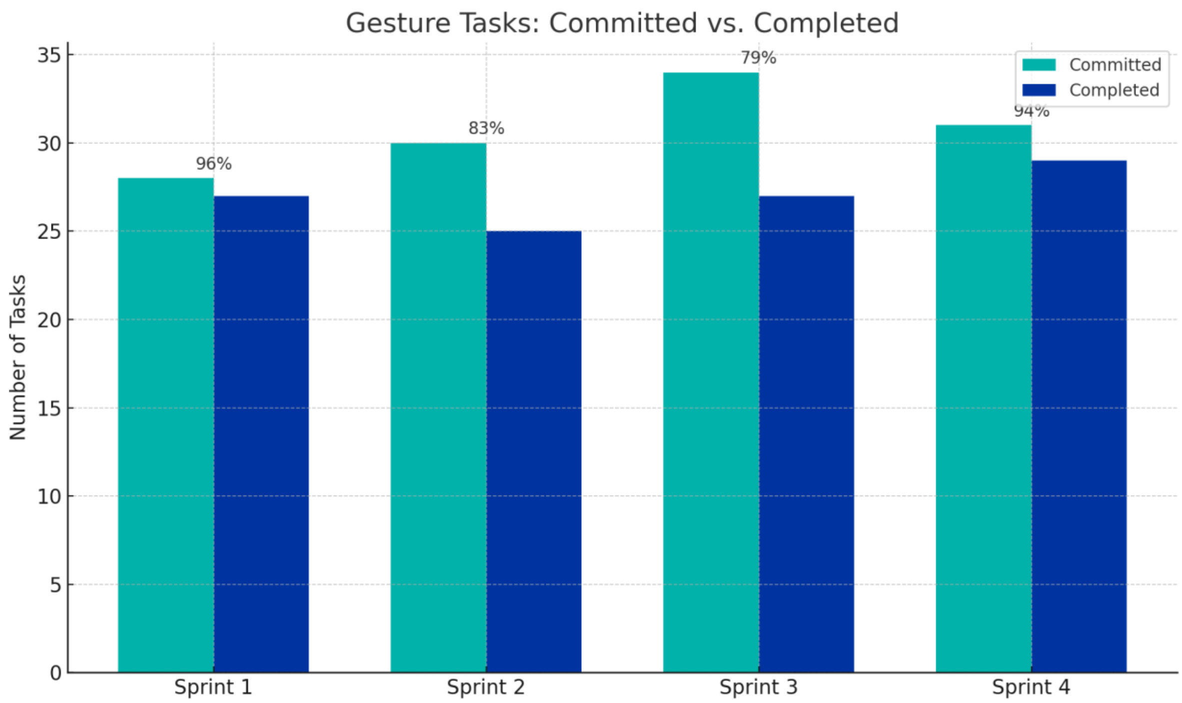


Figure 3.4 Committed vs Completed User Stories

CHAPTER 4

CONCLUSION & FUTURE ENHANCEMENTS

Conclusion

The development of the Hand Gesture-Based Communication System for Disabled Individuals marks a significant advancement in addressing communication barriers for people with speech and hearing impairments. This project successfully combined embedded systems, machine learning, and natural language processing to build a practical, real-time gesture recognition and speech generation tool. Using Raspberry Pi 5 as the core, the system managed camera input, gesture recognition through MediaPipe and LSTM models, and speech synthesis via PicoTTS. It transformed static and dynamic gestures into structured sentences, enhanced by Gemini AI for grammatical refinement and natural voice output. A Flask-based web interface allowed real-time monitoring, sentence editing, and preset customization, while Bluetooth audio streaming (A2DP) and BLE mobile communication extended system accessibility. Extensive testing under different environments ensured stability, accuracy, and low-latency performance. Overall, the project achieved its initial objectives and opened new opportunities for scalable, affordable assistive technologies, presenting a solution that not only improves communication but also promotes empowerment and inclusivity for marginalized communities.

Future enhancement

While the current implementation of the Hand Gesture-Based Communication System for Disabled Individuals meets its core objectives, several future enhancements could significantly improve its adaptability, performance, and user experience. Introducing custom gesture training would allow users to define their own sign vocabulary, making the system more personalized, especially for regional or non-standard sign languages. Adding multilingual support for outputs in languages like Tamil, Hindi, or other dialects would broaden accessibility, complemented by dynamic language switching through the app. Advancing natural language processing with AI models like Gemini or ChatGPT could enable more context-aware sentence generation, making conversations more natural. Future hardware improvements could explore wearable designs, such as smart gloves or wristbands with IMUs and haptic feedback, reducing system bulk and enhancing usability. Integrating visual or tactile feedback mechanisms like LED indicators or vibration motors would confirm gesture recognition effectively in noisy environments. On the software side, implementing cloud-based storage would support cross-device use, remote updates, and caregiver analytics.

APPENDIX

A. PATENT DISCLOSURE FORM

SRM INSTITUTE OF SCIENCE & TECHNOLOGY Deemed to be University u/s 3 of UGC Act, 1956	**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603203**

Application for patent filing

Date	D	D	M	M	Y	Y	Y	Y
	2	9	0	4	2	0	2	5

Name of the Faculty	:	**Dr. R.Babu**
Department	:	**Department of Computational Intelligence**
Faculty ID Number	:	**102893**
Official E-mail ID	:	**babur@srmist.edu.in**
Personal E-mail ID	:	**babu.rajen17@gmail.com**
Contact number:		**7010691102**
Major area of invention	:	A portable, real-time, communication system that translates hand gestures into complete natural language sentences using AI-based gesture recognition and text-to-speech output for disabled individuals.
Narrow focus area of invention	:	Real-time translation of predefined hand gestures into natural sentences and voice output for speech- and hearing-impaired individuals
Title of the invention		Hand Gesture-Based Communication System for Disabled Individual
Earlier status of research	:	Earlier systems relied on expensive hardware, static gesture recognition, and required internet connectivity without offering real-time, natural sentence communication.
How different your invention from similar research / others - **Novelty**?	:	GestureTalk integrates gesture recognition, real-time translation, speech output, and accessibility support within one seamless solution designed for simplicity. Unlike other systems that require multiple tools or

		devices to interpret sign language, our project offers a unified communication platform powered by a modular architecture, enabling instant and accurate gesture-to-speech conversion across all supported sign types simultaneously.
Possible domain for field application	:	● Assistive Communication Technology ● Accessibility Enhancement in Mobile & IoT Devices ● Smart Healthcare Solutions ● Human-Computer Interaction (HCI) ● Education & Awareness for the Disabled Community
Possible sector for commercialization	:	• Individual consumers • Small and medium businesses • Educational institutions • Healthcare organizations
Faculty Signature with date	:	
HoD Remarks / Recommendation for filing patent	:	
Application received by The Review committee on	:	
Review committee remarks	:	

PATENT APPLICATION
Invention Disclosure Form
To be filled by the inventors

Please provide highly relevant information for details asked below and use consistent language while describing the specific feature or element in the invention disclosure.

1. **Title of invention**

   Hand Gesture-Based Communication System for Disabled Individual

2. **Describe the invention**

   GestureTalk is an innovative communication system designed to translate hand gestures into speech for individuals with hearing or speech impairments. The system uses a tag-type device with an ultra-wide camera, coupled with gesture recognition algorithms, to accurately detect sign language. It then converts these gestures into real-time speech output. Built with modular hardware and software architecture, GestureTalk is compact, portable, and accessible, providing a unified assistive solution for the disabled community.

3. Does the invention prov11e a **new use of or improvement to an existing product or process**?

   Yes, GestureTalk offers significant improvements over traditional communication aids. Existing solutions often require specialized gloves or predefined gesture sets and lack real-time conversion. In contrast, GestureTalk enables seamless gesture recognition without wearable devices, offers real-time feedback, and integrates speech output, improving the speed, convenience, and accuracy of gesture-based communication.

4. State the **Novelty** of the invention and specify the claims in the invention

   1. A non-intrusive gesture recognition system that does not require wearables or physical attachments.
   2. Real-time gesture-to-speech translation using camera-based input.
   3. Modular hardware design compatible with Raspberry Pi or Arduino platforms.
   4. Adaptive gesture recognition model that improves accuracy over time based on usage patterns.

5. Describe the **advantages of the present invention over the existing technologies**

   1. User-Friendly: Eliminates the need for users to wear gloves or sensors.
   2. Accessibility: Affordable and designed for easy use by individuals of all ages and technical skill levels.
   3. Real-Time Output: Offers instant gesture recognition and speech conversion.
   4. Customizability: Supports training of new gestures for regional or user-specific sign languages.

6. Describe how the **present invention overcomes the drawbacks** of currently available technology related to your invention.

   1. **Eliminates Wearables**: Many systems rely on specialized gloves or sensors; GestureTalk only requires a camera.
   2. **Improved Real-Time Communication**: Unlike devices with lag or delay, our system focuses on instantaneous conversion.

3. **Scalability and Portability**: Runs on affordable, widely available platforms like Raspberry Pi.
4. **Inclusive Design**: Designed to be intuitive and accessible for disabled individuals and caregivers.

7. Describe **uses, applications and benefits** of the invention.

1. **Personal Communication Aid**: Helps individuals with speech or hearing disabilities communicate effectively in real-time.
2. **Educational Tool**: Useful in special education environments for both teaching and assisting students.
3. **Healthcare Communication**: Enables better communication in hospitals or rehabilitation centers.
4. **Public Service Utility**: Can be used at service counters, transportation hubs, or banks to assist communication.

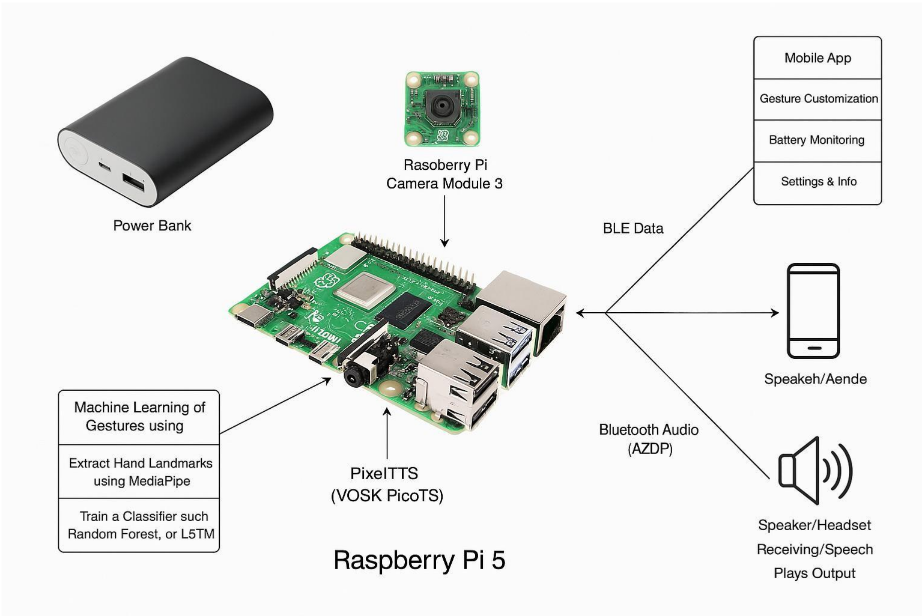
8. Does the focus of the invention results in **societal impact technology**?

GestureTalk addresses social inclusion by empowering disabled individuals with a reliable communication tool. It bridges the communication gap, fosters independence, and enhances confidence. The system contributes to inclusive education, accessible healthcare, and improved quality of life, especially in underserved areas where sign language interpreters are unavailable. It also promotes awareness and empathy towards the disabled community.

9. Characterize the **disadvantages and limitations** of the invention

1. **Environmental Limitations**: Performance may be affected in poor lighting or cluttered backgrounds.
2. **Gesture Variability**: May initially struggle with inconsistent or regional gestures until trained.
3. **Hardware Constraints**: Limited by camera resolution and processing power of embedded systems.
4. **Privacy Considerations**: Continuous camera monitoring raises privacy concerns, requiring transparent policies.
5. **Language Dependency**: Focused on gesture-to-speech in a single spoken language; multilingual support is still under development.

10. Enclose the **sketches, drawings, photographs** and other materials that help in better understating/ illustration of the novelty in the invention.



11 **Current development status of the invention**

A. **Has Your Invention Been Tested Experimentally?** Yes, GestureTalk has been
. tested in a controlled environment using various static and dynamic gestures.

B. **Describe the Experimental Approach of the Invention Also State the Methods Adopted in the Experiment.** GestureTalk was tested using a Raspberry Pi-based prototype equipped with an ultra-wide camera. Tests included detecting standard Indian Sign Language gestures with varying lighting and distances. Users interacted with the system for real-time gesture-to-speech output, and performance was logged for accuracy, latency, and usability.

C. **Are the Experimental Data Documented in a Formal Log or Any Instrumental Confirmation Available for the Invention?** Yes, test logs, performance metrics, user accuracy reports, and hardware setup documentation are maintained and available in a shared research repository.

D. **Is Further Development of Your Invention Necessary or Development of the Invention is in Progress?** Yes, development is ongoing. Planned enhancements include:

- Deep learning model integration for adaptive recognition
- Multilingual speech support
- Mobile app extension
- Power optimization for longer battery life
- Voice tone customization for different user preferences

13. **INVENTOR(S) AND/OR CONTRIBUTOR(S):**

	INVENTOR (1)		INVENTOR (2)
Name:	Jeevan Baabu Murugan		Surya Sathyanarayanan
Address:	Department of, CINTEL SRM IST, Kattankulathur campus-603203		Department of CINTEL, SRM IST, Kattankulathur campus-603203
City and State:	Chennai &Tamilnadu		Chennai &Tamilnadu
Citizenship (Country):	INDIAN		INDIAN
Email Id: (official and personal ID is required)	jm7531@srmist.edu.in jeevanbaabu03@gmail.com		ss7626@srmist.edu.in surya.sdp2004@gmail.com
Contact number:	8438844636		8838335431

B. SAMPLE CODING

```
import cv2

import numpy as np

import mediapipe as mp

import pyttsx3

import google.generativeai as genai

from tensorflow.keras.models import load_model

import time

import os

import json

from dotenv import load_dotenv

import requests

load_dotenv()

GOOGLE_API_KEY = os.getenv("GOOGLE_API_KEY")

genai.configure(api_key=GOOGLE_API_KEY)

model_gemini = genai.GenerativeModel(model_name="gemini-1.5-pro-latest")

pose_model = load_model("pose_classifier_final.h5")

pose_labels = np.load("pose_labels_final.npy")

gesture_model = load_model("preset_gesture_model.h5")

gesture_labels = np.load("preset_gesture_labels.npy")

PRESETS_FILE = "presets.json"

PRESET_UPDATE_FLAG = "preset_update.flag"

last_preset_check = time.time()
```

```
def load_presets():
        if not os.path.exists(PRESETS_FILE):
          presets = {}
          for i, label in enumerate(gesture_labels):
            if label.startswith("preset"):
              presets[label] = label
          if not presets:
            presets = {
              "preset1": "hello",
              "preset2": "thank you",
              "preset3": "help me"
            }
          with open(PRESETS_FILE, 'w') as f:
            json.dump(presets, f)
        else:
          with open(PRESETS_FILE, 'r') as f:
            presets = json.load(f)
return presets
      presets = load_presets()
      tts = pyttsx3.init()
      tts.setProperty('rate', 160)
```

```
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(static_image_mode=False, max_num_hands=2)
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT,  480)
cap.set(cv2.CAP_PROP_FPS, 30)  # Try to set higher FPS
last_prediction = ""
cooldown_counter = 0
cooldown_frames = 30  # Reduced from 60 to improve speed
detection_delay = 10  # Reduced from 20
WEB_SERVER_URL = "http://localhost:5000"
def extract_landmarks(frame):
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = hands.process(rgb)
    if result.multi_hand_landmarks:
        full = []
        for i in range(2):
            if i < len(result.multi_hand_landmarks):
                full.extend([c for lm in result.multi_hand_landmarks[i].landmark for c in (lm.x,
lm.y, lm.z)])
            else:
                full.extend([0.0] * 63)
```

```
        return np.array(full)
    return None
detection_history = []
HISTORY_SIZE = 5
STABILITY_THRESHOLD = 0.7
last_sent_prediction = ""
frame_counter = 0
def update_preset_mapping():
    global preset_mapping
    preset_mapping = {}
    for preset_key, preset_value in presets.items():
        # Find the index of this preset in gesture_labels
        if preset_key in gesture_labels:
            preset_mapping[preset_key] = preset_value
    print("Updated preset mappings:", preset_mapping)
update_preset_mapping()
print("Available gesture labels:", gesture_labels)
while True
    current_time = time.time()
    if current_time - last_preset_check > 2:
        last_preset_check = current_time
```

```
        # Check if the update flag file exists
        if os.path.exists(PRESET_UPDATE_FLAG):
            try:
                # Load updated presets
                presets = load_presets()
                update_preset_mapping()
                print("Presets reloaded from file:", presets)
                os.remove(PRESET_UPDATE_FLAG)
            except Exception as e:
                print(f"Error reloading presets: {e}")
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)
    frame_counter += 1
    if frame_counter % 2 != 0:
        cv2.imshow("Sign Detection", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
break
continue
    landmark = extract_landmarks(frame)
    if landmark is not None:
```

```
            norm = (landmark - np.mean(landmark)) / (np.std(landmark) + 1e-6)
            pose_pred = pose_model.predict(np.expand_dims(norm, axis=0), verbose=0)[0]
            pose_word_idx = np.argmax(pose_pred)
            pose_word = pose_labels[pose_word_idx]
            pose_conf = np.max(pose_pred)
            gesture_pred = gesture_model.predict(np.expand_dims(norm, axis=0), verbose=0)[0]
            gesture_idx = np.argmax(gesture_pred)
            gesture_label = gesture_labels[gesture_idx]
            gesture_conf = np.max(gesture_pred)
            word = ""
            if gesture_conf > 0.95 and gesture_label in preset_mapping:
                # Use the preset mapping to get the user-defined value
                word = preset_mapping[gesture_label]
                cv2.putText(frame, f"Preset: {gesture_label} -> {word}", (10, 60),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            elif pose_conf > 0.90:
                word = pose_word
            if word:
                detection_history.append(word)
                if len(detection_history) > HISTORY_SIZE:
                    detection_history.pop(0)
                if len(detection_history) == HISTORY_SIZE:
most_common = max(set(detection_history), key=detection_history.count)
```

```
frequency = detection_history.count(most_common)
            if frequency >= STABILITY_THRESHOLD and most_common !=
    last_sent_prediction:
                last_prediction = most_common
                last_sent_prediction = most_common
                # Update the web interface
                try:
                    requests.post(f"{WEB_SERVER_URL}/update_word", json={"word":
    last_prediction}, timeout=0.5)
                except (requests.exceptions.RequestException, requests.exceptions.Timeout):
                    pass
                        cv2.putText(frame, f"Detected: {last_prediction}", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        cv2.imshow("Sign Detection", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

C. PLAGIARISM REPORT