International Journal of Engineering Research and Development
e- ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 22, Issue 8 (August 2026), PP 18-22

Responsive Web Application for Children to Learn Sign Language

Prajwal V D[1], Dr. Swamy L N[2], Mr. Tammarguddi Mahmad Husen[3], Mrs. Surabhi M V[4], Dr. Sukruth Gowda M A[5]

[1]*Student, Department of CSE (MCA Programme), VTU CPGS Mysuru-570029.*
[2]*Assistant Professor, Department of CSE (MCA Programme), VTU CPGS Mysuru-570029.*
[3,4]*Research Scholar, Department of CSE (MCA Programme), VTU CPGS Mysuru-570029.*
[5]*Assistant Professor, Department of Computer Science and Engineering, Presidency University, Bangalore.*

ABSTRACT
In today's digital world, people expect web applications to be easy to use, accessible, and inclusive for everyone. But traditional text-based interfaces can be limiting especially for individuals with hearing, speech, or physical challenges making it harder for them to interact with technology. To bridge this gap, speech-based interaction has become a powerful tool, allowing users to communicate naturally through spoken language. This project builds a web-based Speech-to-Text (STT) and Text-to-Speech (TTS) system to bridge communication gaps. Users speak to generate text or type to trigger synthetic audio. We embedded the tool directly into the Vue.js navigation bar. It works fast. A Python Flask server manages the backend logic. When the frontend transmits text, the backend invokes synthesis libraries to produce clear audio files. This helps users with visual impairments or literacy challenges. HTTP requests handle the data flow. The connection remains stable. The system applies core concepts like RESTful APIs and asynchronous data exchange. It proves that Vue.js and Python can integrate to solve accessibility hurdles. We prioritized inclusive design. This ensures the app serves diverse user needs without friction.

Keywords: *Sign Language, Interactive Learning, Speech-to-Text, Text-to-Speech, Web Application*

Date of Submission: 25-07-2026 Date of acceptance: 06-08-2026

I. INTRODUCTION

Communication is a fundamental requirement for human interaction, learning, and social participation. Hearing or speech impairments often hinder standard tech interactions. Modern speech processing and web tools now bridge these gaps effectively. We built a web-based STT and TTS system to foster inclusion. It works. Users toggle between vocalizing text and hearing typed words via a Vue.js interface.Browser APIs and lightweight backends have replaced costly, specialized hardware. This project leverages these modern stacks to create a practical, portable communication tool. Integration is straightforward.

Accessibility remains our primary objective. Adding voice-driven features assists users with physical or cognitive disabilities. It also enables hands-free operation for general users. Utility increases for everyone.We utilized a client-server architecture to ensure scalability. Vue.js manages the responsive frontend and captures browser-based audio. Meanwhile, a Python Flask backend handles the heavy TTS processing. This separation keeps the codebase clean. It stays organized.This tech mirrors logic found in virtual assistants and modern support platforms. Developing it provides direct experience with real-time API communication and UI design.

The final product emphasizes extreme simplicity. A single button triggers the recognition engine. This removes friction for the end-user. Similarly, users can enter text manually and trigger speech output without navigating away from the page. This seamless experience ensures that the tool remains intuitive even for users with minimal technical knowledge.In an academic context, this project demonstrates the application of theoretical concepts such as web architecture, speech processing fundamentals, and software integration. It also highlights the importance of designing inclusive technologies that provide the user needs. Overall, the project serves as a meaningful example of how modern web technologies can be combined to create practical solutions that improve accessibility and communication.

II. LITERATURE REVIEW

The literature survey forms an essential part of any academic project, as it provides a comprehensive understanding of existing research, technologies, and methodologies related to the problem domain. Our literature survey tracks the evolution of STT and TTS systems from rigid beginnings to modern, flexible

implementations. Analyzing past research reveals what worked and where gaps remain. This data grounds our project.Early speech recognition relied on rule-based logic and extensive user training. These systems struggled with accuracy. However, they introduced vital concepts like acoustic modeling and phonetics. We still face challenges like background noise and dialect variations today. They never truly go away.The shift toward Hidden Markov Models (HMMs) and probabilistic methods marked a leap in performance. These statistical approaches required massive datasets and high computational power. They often failed outside of controlled lab settings. We learned that accuracy often competes with system speed.

1. Web-Based Speech Recognition Systems

Native browser APIs now handle recognition directly on the client side. This removes the need for heavy server processing. It makes apps faster. Developers prefer this for lightweight accessibility tools. However, browser engines vary in quality. Results aren't always consistent across different platforms.

2. Speech-to-Text Technologies for Accessibility

STT systems provide a vital communication link for those with hearing or vocal impairments. Real-time transcription transforms classrooms and office environments into inclusive spaces. Transcription accuracy has peaked recently. Still, heavy background noise and regional accents can cause errors. Clear audio remains essential.

3. Text-to-Speech Systems in Human–Computer Interaction

TTS allows hardware to communicate vocally with users. This is a game-changer for visually impaired individuals. Research proves that natural, human-like voices increase user trust and engagement. Robotic tones fail in educational settings. We prioritized vocal fluidity. It feels more personal.

4. Python-Based Speech Processing Applications

Python has become a popular choice for building speech-related tools because it’s simple to use and has strong library support. Studies show that Python frameworks make it easy to quickly develop and connect speech systems with web technologies. The trade-off is that developers often need to fine-tune performance to handle real-time speech tasks smoothly.

5. Flask Framework for Lightweight Backend Services

Flask has been frequently cited in literature as an effective microframework for developing RESTful APIs. Research indicates that Flask is suitable for small to medium-scale applications due to its flexibility and minimal overhead. Its simplicity makes it ideal for academic projects and prototype development.

6. Client–Server Architecture in Web Applications

Many researchers highlight the importance of client–server architecture in modern web applications. This model improves system modularity and scalability by separating frontend and backend responsibilities. Literature confirms that such architecture enhances maintainability and allows independent updates of system components.

7. Voice-Based Interfaces in Education Systems

Studies show that voice-based interfaces improve learning accessibility for students with disabilities. Speech-enabled systems support inclusive education by allowing alternative input and output methods. Research also suggests that such systems improve engagement and reduce dependency on traditional input devices.

8. Real-Time Audio Processing Challenges

Literature identifies real-time audio processing as a technically demanding task. Researchers note that latency, noise interference, and hardware limitations significantly impact system reliability. Effective error handling and user feedback mechanisms are recommended to mitigate these challenges.

9. Browser APIs for Speech Recognition

Several studies analyze browser-provided speech recognition APIs as cost-effective solutions for speech input. These APIs reduce development complexity and hardware dependency. However, literature highlights concerns regarding privacy, browser support limitations, and inconsistent performance across platforms.

10 Accessibility Focused Wef Application Design

Research emphasizes the importance of accessibility in web application design. Speech-based features are identified as key accessibility tools. Studies recommend simple interfaces, clear feedback, and minimal interaction steps to improve usability for differently-abled users.

III. METHODOLOGY

The development of the Speech-to-Text and Text-to-Speech system follows a structured and systematic methodology to ensure reliability, usability, and scalability. The project was developed step by step, with each phase focusing on a specific part of the system.

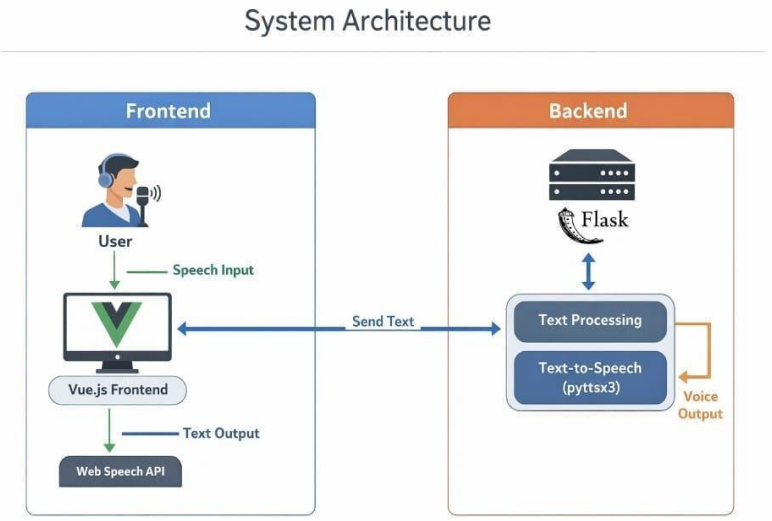

Fig: 3.1: System Architecture

The system architecture of the Speech-to-Text and Text-to-Speech web application is designed using a client–server model, which ensures clear separation of responsibilities, scalability, and ease of maintenance. Our system integrates a Vue.js frontend with a Python Flask backend. This pairing ensures efficient communication and simplifies future scaling. The architecture uses three distinct tiers.

1. Presentation Layer (Frontend)

Built with Vue.js, HTML, and CSS, this layer manages the user experience. Users access the tool via the navigation bar. It features dedicated buttons for speech recognition and fields for text input.We utilized the Web Speech API to handle recognition at the browser level. When a user speaks, the browser converts audio to text locally. This strategy eliminates server lag. Transcription appears instantly.

2. Application Layer (Communication)

This tier acts as the bridge. It facilitates data exchange using RESTful APIs via HTTP. For instance, the frontend transmits typed text to the backend through a POST request. It ensures all data remains structured and reliable during transit.

3. Processing Layer (Backend)

The Python Flask backend provides a lightweight environment for heavy logic. It hosts API endpoints that receive text from the frontend. Once triggered, the backend invokes synthesis libraries to generate audio.This layered approach keeps the system modular. You can update the UI without touching the backend logic. It also allows us to add features like user authentication later without a total redesign. The system staysflexible.

IV. RESULTSAND DISCUSSION

Fig: 4.1 Home Page

The Home page serves as the entry point for the Interactive Sign Language Application. It introduces the mission immediately. This section focuses on bridging communication gaps for the deaf and hard-of-hearing community through accessible learning tools.
A navigation bar directs users to Home, Contents, Speech Tool, Contact Us, About Us, Register, and Login. It works fast. The layout features a prominent banner with hand sign graphics to engage visitors and build instant awareness. Users start here to explore the system's core features.

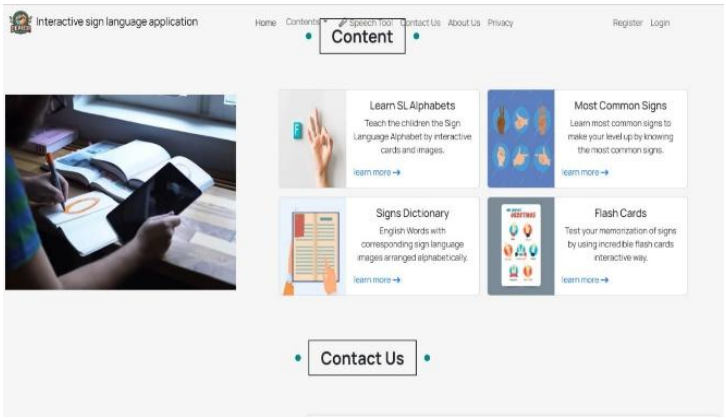

Fig: 4.2 Content Page

The Contents page hosts the application's primary learning modules. It streamlines navigation. This central hub lets users jump directly into specific sign language training segments based on their current needs.
The page features these specific tools:

- **Learn Sign Alphabets:** Users master individual letters through high-resolution visual aids. It builds the basics.
- **Most Common Signs:** This section focuses on gestures used in daily interactions.
- **Signs Dictionary:** This tool maps English vocabulary to specific sign imagery. It is a quick reference.
- **Flash Cards:** An interactive module built for memory retention and active practice.

The design ensures learners find their preferred lesson without delay. Choice is key

Fig: 4.3 Learning Sign Alphabet Page

This module teaches the complete A–Z sign language manual alphabet. It starts with basics. Each letter features specific images or video clips demonstrating precise hand positioning and movement.The interface targets children and novice learners through a structured, step-by-step methodology. Visual cues drive the experience. Using these graphic demonstrations speeds up memory retention and helps users master finger-spelling quickly.

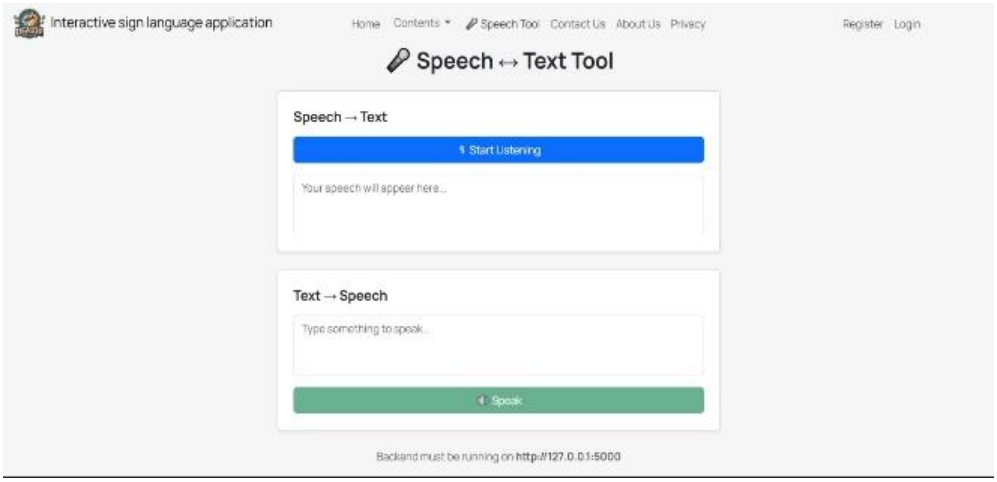

Fig: 4.4 Speech to Text Tool Page

The Speech to Text Tool translates spoken words into written format in real-time. It bridges the gap. Users initiate the process by clicking the "Start Listening" button and speaking directly into their device's microphone.The system processes the audio signal and renders it as legible text on the interface. Accuracy matters. This functionality enables deaf users to follow oral conversations and fosters seamless interaction between hearing and non-hearing individuals.

V. CONCLUSION

The development of the web-based Speech-to-Text (STT) and Text-to-Speech (TTS) system represents an important step toward improving digital accessibility and human-computer interaction. This project successfully demonstrates how modern web technologies can be empowered create an efficient and user-friendly communication tool. By leveraging Vue.js for frontend development and Flask for backend processing, the system achieves a clean and modular architecture.The system effectively converts spoken language into readable text and transforms typed text into audible speech. These core functionalities address communication challenges faced by individuals with speech or hearing impairments. Additionally, the system benefits general users by enabling hands-free interaction and faster communication. The simplicity of the interface ensures ease of use across different user groups.

The project also highlights the importance of client-server architecturein modern web applications. The clear separation between frontend and backend components improves maintainability and scalability. This architectural approach allows future developers to enhance individual modules without affecting the entire system.Performance and usability were given significant attention during development. The system delivers fast response times and clear audio output, ensuring a smooth user experience. Basic error handling makes the system more reliable and resilient. This ensures it can be used confidently in both educational and assistive settings without frequent disruptions.Privacy and security have also been carefully considered. The system avoids storing unnecessary audio or text data, which helps protect user privacy. In addition, browser permission controls give users full authority over microphone access.

REFERENCES

[1]. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2023.
[2]. S. Young, G. Evermann, and M. Gales, “The HTK book: Speech recognition toolkit,” Cambridge University Engineering Dept., Cambridge, U.K., Tech. Rep., 2021.
[3]. M. Stone and A. Smith, “Web-based speech recognition systems using browser APIs, vol. 14, no. 2, pp. 95–108, 2022.
[4]. A. Kumar and R. Verma, “Speech-to-text applications for accessibility,” International Journal of Assistive Technologies, vol. 8, no. 1, pp. 23–31, 2021.
[5]. P. Taylor, “Text-to-speech synthesis: A review,” IEEE Signal Processing Magazine, vol. 26, no. 3, pp. 42–52, May 2020.
[6]. J. Allen, “Human–computer interaction through speech interfaces,” Communications of the ACM, vol. 63, no. 4, pp. 68–76, Apr. 2020.
[7]. M. Grinberg, Flask Web Development: Developing Web Applications with Python. Sebastopol, CA, USA: O’Reilly Media, 2022.
[8]. A. Grigorik, “High-performance client–server architecture for modern web systems,” IEEE Internet Computing, vol. 24, no. 1, pp. 30–38, Jan.–Feb. 2020.
[9]. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Boston, MA, USA: Addison-Wesley, 2019.
[10]. R. W. Picard, “Affective computing and speech interaction,” IEEE Transactions on Affective Computing, vol. 11, no. 1, pp. 1–13, Jan. 2020.