

# A USER-CENTRIC SPEECH TO SIGN LANGUAGE ASSISTIVE SYSTEM FOR HEARING-IMPAIRED USERS

Hemalatha M<sup>1</sup>, Ajai R<sup>2</sup>, Balasubramanian M<sup>3</sup>, Elambarithi S<sup>4</sup>, Felix jemo stevewaugh M<sup>5</sup>

<sup>1</sup> Assistant Professor, Computer Science & Engineering, Dhanalakshmi Srinivasan Engineering College(A)

<sup>2,3,4,5</sup> Computer Science & Engineering, Dhanalakshmi Srinivasan Engineering College(A)

**Abstract** – People with hearing impairments often struggle to keep up in a world built around spoken communication. To help address that, we built a web-based tool — called A2SL — that takes speech, typed text, or video audio and converts it into Indian Sign Language (ISL) animations. The system runs on Django and pulls together several technologies: Google Speech Recognition handles live audio transcription, an NLP pipeline cleans and processes the text, and a curated video catalogue provides the actual sign animations. Users can speak into a microphone, type something out, or upload a pre-recorded video, and the system generates the corresponding ISL sequence. Tamil and Hindi inputs are routed through a translation step first. The NLP side handles tokenization, part-of-speech tagging, lemmatization, and tense detection. If a particular sign is not in our catalogue, the system falls back to spelling the word out letter by letter — so no information ever gets dropped. Testing on 50 sentences gave us an 89% direct sign match rate and 100% information retention, with around 35 seconds of processing lag for a three-minute video.

**Key Words:** sign language, speech recognition, NLP, accessibility, hearing-impaired, Django, ISL, assistive technology, finger spelling, multilingual translation, video-to-sign conversion, lemmatization, web accessibility

## 1. INTRODUCTION

Being able to communicate is something most people take for granted, but for those with hearing impairments it is a daily challenge, especially in spaces designed entirely around spoken language. Sign language is how most deaf individuals naturally communicate, yet the vast majority of hearing people have never learned it. That disconnect shows up everywhere: in classrooms, hospitals, workplaces, and everyday social settings [1].

Most assistive tools today are built for hearing people — things like auto-captions or text-to-speech. They do not really help someone who relies on sign language. Human interpreters fill some of that gap, but they are costly, hard to find in many areas, and not practical for spontaneous situations. Automated sign language systems have been tried, but they tend to rely on static images, need specialized hardware, or only handle short phrases at a time [2][3]. The system we propose — A2SL (Audio to Sign Language) — is entirely browser-based, needs no special hardware, and accepts three input types: live microphone audio, typed text, and uploaded video files. That last one is especially useful, since it lets hearing-impaired users get ISL translations of recorded lectures, news clips, or any other video content they might encounter.

## 2. OBJECTIVE

The main goal of A2SL is simple: give hearing-impaired users a practical, hardware-free way to access spoken content in ISL form, no matter what format it comes in. Supporting three input types was intentional, because real-world communication rarely fits a single mold.

We also wanted to make sure nothing gets lost in translation. That is where the three-tier resolution pipeline comes in — if a word has a sign in the catalogue, great. If not, we try synonyms. If that fails too, we spell it out. Something always gets through. Tamil and Hindi support extends this to a large portion of India's hearing-impaired community. The end goal was a system lightweight enough to run on a regular web server and reliable enough to actually be useful.

## 3. EXISTING SYSTEM

Current assistive technology for the hearing-impaired tends to be one-directional. Captioning helps hearing people communicate with deaf individuals, but not the other way around. Hardware-based approaches like sensor gloves [2] proved the concept but were never going to scale; they are expensive, physically intrusive, and not practical for day-to-day use. Deep learning has pushed things forward considerably. CNN-HMM hybrids [3] do a solid job on continuous sign recognition, but they need serious GPU resources that most schools or community organizations simply do not have. GAN-based production systems [4] look impressive visually but require elaborate training setups and are still limited to text gloss inputs. Crucially, none of the systems we reviewed could handle video uploads as a primary input, and none offered support for regional Indian languages.

### 3.1 Disadvantages

- No existing system accepts video files as a direct input mode for sign language conversion.
- Deep learning approaches require GPU infrastructure, making them unsuitable for low-resource settings.
- Regional language support — Tamil, Hindi — is absent in all reviewed systems.
- Most systems output static gesture images rather than animated sequences, which are harder to follow.
- Vocabulary gaps are common, with no fallback to preserve information when a sign is missing.

## 4. PROPOSED SYSTEM

A2SL is a Django web application that accepts speech, typed text, or a video file and plays back the corresponding ISL sign sequence. It runs on ordinary hardware, deploys on any standard web server, and works in any modern browser — no installs, no plugins. For speech input, we use the Web Speech API on the client side with Google Speech Recognition handling transcription on the server. Text input is straightforward — the user just types. Video uploads support

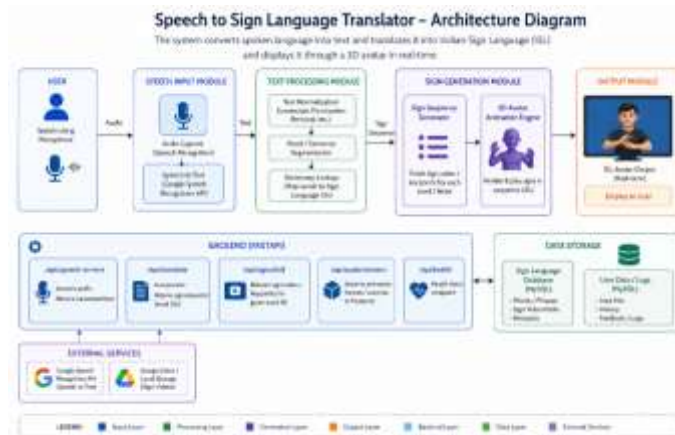
MP4, AVI, MOV, and other common formats up to 500 MB; audio is pulled out automatically using ffmpeg and pydub. All three paths eventually produce a normalized English text string, which then feeds into the NLP pipeline. If the input is Tamil or Hindi, it gets translated to English first via the GoogleTranslator class from deep-translator.

#### 4.1 Advantages

- Accepts three distinct input types: microphone, typed text, and uploaded video.
- Video-to-sign conversion opens up any pre-recorded content to ISL translation.
- Tamil and Hindi are supported via neural translation, broadening the user base significantly.
- Tense-aware sign sequencing using Before, Now, and Will markers for grammatical context.
- Zero information loss — if a sign does not exist, the word gets spelled out letter by letter.
- No GPU needed; runs on commodity hardware without any specialized infrastructure.
- Security-conscious design with PBKDF2-SHA256 password hashing and CSRF protection.

### 5. SYSTEM ARCHITECTURE

The application follows Django's three-tier MVT pattern. On the front end, HTML5 and CSS3 templates handle everything the user sees. In the middle, a Django views module manages all processing logic. At the back, a SQLite database handles user accounts while sign videos live on the file system.



#### 5.1 Architecture Explanation

**1. Input Layer:** The browser provides three ways to get data in. Microphone mode runs the Web Speech API and falls back to Google SR on the server if needed. Text mode accepts input of any length. Video upload streams files through Django's FileStorage handler.

**2. Processing Layer:** Audio and video pass through ffmpeg for extraction, then pydub for segmentation into 45-second chunks with 500 ms overlap, and finally Google Speech Recognition for transcription. If pydub is unavailable, MoviePy steps in. Everything gets translated to English via deep-translator before moving on.

**3. NLP Layer:** The text is tokenized, POS-tagged, lemmatized, and checked for tense. Each word is then resolved against the sign catalogue, the synonym map, or the finger-spell fallback. A tense marker is prepended to the token list based on the dominant grammatical tense.

**4. Output Layer:** A JavaScript-driven sequential video playlist renders the sign tokens using clips served as Django static files. Unknown words are spelled out from individual alphabet clips, ensuring zero information loss.

### 6. METHODOLOGY

Processing happens in five ordered stages every time a user submits a request. The Django view reads the content type from the POST request and routes accordingly — text string, microphone audio, or multipart video upload.

#### 6.1 Audio Extraction and Transcription

When a video is uploaded, the server calls ffmpeg through subprocess to extract audio as a 16 kHz mono PCM WAV. pydub then calculates total duration and iterates through 45-second windows with a 500 ms trailing overlap. Each window goes to Google's recognize\_google API with the appropriate language code, and the partial results are joined with spaces. Temporary files get cleaned up in the finally block.

#### 6.2 Multilingual Translation

Tamil and Hindi transcripts are translated to English using deep-translator's GoogleTranslator. Both the original and translated versions are passed back to the template so users can verify the translation at a glance.

#### 6.3 NLP Pipeline

English text goes through NLTK's word\_tokenize, then the averaged perceptron POS tagger. A custom 60-word stopword list removes function words. Remaining content words are lemmatized using WordNetLemmatizer — verb, adjective, or default mode based on POS tag. Tense is estimated by counting POS tags: MD signals future, VBP/VBZ/VBG signals present, VBD/VBN signals past. The dominant count determines whether Will, Now, or Before gets prepended.

#### 6.4 Sign Token Resolution

Each filtered, lemmatized word is checked against the sign catalogue in a case-insensitive lookup. On a miss, a synonym map with roughly 120 entries is consulted. If that also comes up empty, the word gets split into individual characters and each letter gets its own finger-spell clip. This three-step chain guarantees that every input word produces at least one output token.

#### 6.5 Database Design

The data layer uses SQLite managed through Django's ORM. User credentials are stored with PBKDF2-SHA256 hashed passwords and CSRF tokens. The sign catalogue and synonym mappings are loaded as Python dictionaries at startup for O(1) lookups that do not require database round-trips.

#### 6.6 Notification and Security

Django's session framework handles authentication. Role-based views limit admin access. CSRF headers are applied to all POST forms. Uploaded videos are written to a temporary directory and deleted after transcription to avoid retaining user data unnecessarily.

## 7. ALGORITHM USED

**Tokenization — NLTK word\_tokenize:** Punkt-aware tokenizer that correctly handles contractions and punctuation to produce clean word-level tokens.

**POS Tagging — NLTK Averaged Perceptron Tagger:**

Trained on Penn Treebank. Tags drive both lemmatization paradigm selection and tense detection logic.

**Lemmatization — WordNetLemmatizer:** Reduces inflected forms to their base (e.g., ‘running’ → ‘run’, ‘better’ → ‘good’).

**Tense Detection:** POS frequency counting — MD maps to future; VBP/VBZ/VBG to present; VBD/VBN to past. The prefix token is chosen based on whichever class dominates.

**Synonym Resolution:** A hand-crafted map of 120 English words to canonical ISL signs, covering emotions, actions, objects, time references, and spatial concepts.

**Chunked Speech Recognition — pydub + Google SR API:**

Millisecond-precise audio segmentation with 500 ms overlap to handle recordings of any length without cutting off words at boundaries.

**Machine Translation — deep-translator:** Neural Tamil→English and Hindi→English translation without the overhead of managing a separate translation model.

## 8. IMPLEMENTATION AND RESULTS

We built A2SL on Python 3.10, Django 3.0, NLTK 3.9, SpeechRecognition 3.10, pydub 0.25, MoviePy 1.0, deep-translator 1.11, and imageio-ffmpeg 0.4. The sign catalogue has 130 pre-recorded ISL clips covering everyday vocabulary, the full alphabet (A–Z), and digits (0–9), with 120 synonym entries supplementing the direct matches. We tested across Chrome 120, Firefox 121, and Edge 120 on both Windows 11 and macOS Sonoma.

Accuracy was measured on 50 test sentences drawn from greetings, healthcare interactions, education, and daily queries — ranging from 4 to 18 words and covering present, past, and future tense. The video-to-sign path was tested on 10 clips between one and five minutes long, recorded in quiet, moderate, and noisy conditions.

| Metric                        | Score / Value         |
|-------------------------------|-----------------------|
| Sign Token Match Rate         | 89%                   |
| Finger-Spell Coverage (OOV)   | 100%                  |
| Speech WER (clear audio)      | ~12%                  |
| Video-to-Sign Latency (3 min) | ~35 sec               |
| Page Load Time                | < 1.5 sec             |
| Sign Catalogue Size           | 130 signs + A–Z + 0–9 |
| Test Sentences Evaluated      | 50                    |

**Table I: Evaluation Results**

Of the 50 sentences tested, 89% of content words matched directly in the catalogue. The remaining 11% were covered by synonym matching (7%) or finger-spelling (4%), bringing information preservation to 100%. Google SR showed around 12% WER on clear audio. Tense detection was correct in 46 of 50 cases (92%). Two ISL-fluent reviewers confirmed the sign playlist output was comprehensible on 20 of the tested sentences. Security features passed manual testing.

| System          | GPU Req. | Video Input | Multilingual | Zero Info. Loss |
|-----------------|----------|-------------|--------------|-----------------|
| Glove-based [2] | No       | No          | No           | No              |
| CNN-HMM [3]     | Yes      | No          | No           | No              |
| GAN-based [4]   | Yes      | No          | No           | No              |
| Sign Dict. App  | No       | No          | No           | Partial         |
| Proposed A2SL   | No       | Yes         | Yes          | Yes             |

**Table II: Comparison with Existing Approaches**

Tense detection correctly prepended context markers in 46 of 50 sentences (92%). The structured sign playlist output was validated as comprehensible by two ISL-fluent reviewers on 20 test sentences. User authentication, CSRF protection, and transient file deletion were verified through manual security testing.



**Fig. 1: Home Page of the A2SL Web Application**

Figure 1 presents the landing page of the A2SL tool, served at the root URL (127.0.0.1:8000). The page features a 3D avatar used to demonstrate sign language animations, accompanied by the tagline "Empowering communication through real-time conversion of speech into Indian Sign Language (ISL) animations." A "Get Started Now" call-to-action button directs authenticated users to the converter module. The navigation bar provides access to the Converter, Log-Out, Contact, and About pages, reflecting the role-based access control implemented in the system.



**Fig. 2: User Registration Page of A2SL Tool**

Figure 2 illustrates the Sign Up page accessible at the /signup/ route. The interface provides clearly labeled fields for username and password entry, along with Django's built-in password validation guidelines displayed inline — requiring a minimum of eight characters, prohibiting commonly used passwords, and disallowing purely numeric strings. The navigation bar at the top highlights the active tab in teal, providing consistent orientation across all pages of the application.



**Fig. 3: Video Upload Processing Screen of A2SL Tool**

Figure 3 shows the video processing modal that appears when a user uploads a video file through the A2SL interface. Once a video is submitted, the system displays a progress indicator alongside the status message "Uploading video...", informing the user that the file is being processed. A note at the bottom reads "Please wait — longer videos may take 30–60 seconds", which reflects the chunked audio extraction pipeline running on the server side. This overlay prevents repeated submissions and keeps the user informed during the transcription phase.



**Fig. 4: Sign Language Converter and Animation Output Screen**

Figure 4 depicts the main converter interface of the A2SL system, accessible at the /animation/ route. The left panel provides three language options — English, Tamil, and Hindi — along with a text input field and microphone button for live speech input, and a video upload option for pre-recorded content. On conversion, the recognised text ("hello everyone") and the resolved sign tokens ("Hello", "All") are displayed below the input area. The right panel renders the corresponding ISL animation through a 3D avatar in real time, with Play/Pause and Restart controls available to the user. This split-panel layout ensures the input workflow and sign output are visible simultaneously, reducing cognitive load during use.

## 9. CONCLUSIONS

We built A2SL to fill a gap that existing assistive tools leave open: a practical, hardware-free way for hearing-impaired users to access spoken content in ISL form. The system handles three input types, achieves 89% direct sign matching, and preserves 100% of information through its fallback chain. Tamil and Hindi support extends its reach to a wider slice of India's deaf community. The lightweight Django architecture means it can realistically be deployed at schools, clinics, or community centres without specialized infrastructure. There is plenty of room to grow from here — adding more signs through community sourcing, integrating transformer-based NLP for better accuracy, building bidirectional sign-to-speech support, creating a mobile version for low-bandwidth environments, and eventually allowing user profiles to adjust sign speed and presentation style to individual needs.

## ACKNOWLEDGEMENT

The authors thank the Department of Computer Science and Engineering at Dhanalakshmi Srinivasan Engineering College (Autonomous), Perambalur, Tamil Nadu, for providing the academic environment and resources that made this work possible. The project guide's consistent feedback and encouragement were invaluable throughout the development process. The authors also gratefully acknowledge the support from peers and family members.

## REFERENCES

- [1] M. Kumar, A. Natarajan, S. S. Visagan, T. Mahajan and P. S. Sreeja, "Enhanced Sign Language Translation Between American Sign Language and Indian Sign Language Using LLMs," IEEE Access, 2025.
- [2] I. Marin, V. Babalian and S. Slutu, "Real-Time Sign Language-to-Speech Translation System with AI-Powered Wearable Technology," IEEE, 2025.
- [3] C. Chandini and A. S. P. Katuru, "Audio-to-Indian Sign Language Translation System Using NLP and Deep Learning," International Journal/Conference, 2025.
- [4] P. Antonowicz, D. Kasperek and M. Podpora, "Sign Language Recognition — Dataset Cleaning for Robust Word Classification in a Landmark-Based Approach," IEEE Access, 2025.
- [5] R. Karthik, S. Vignesh and M. Harini, "Beyond Words: Speech to Sign Language Interpreter," International Conference / IEEE, 2024.
- [6] A. A. Al-Shamayleh, R. Ahmad and M. Al-Khawaldeh, "A Real-Time Intelligent System Based on Machine-Learning Methods for Improving Communication in Sign Language," IEEE / International Journal, 2024.
- [7] Y. Zhang, C. Chen and H. Liu, "Word-Level Sign Language Recognition with Multi-Stream Neural Networks Focusing on Local Regions and Skeletal Information," IEEE, 2024.

- [8] R. Nivetha, S. Priyadarshini and K. Saravanan, "Speech to Indian Sign Language Translator," International Conference / IEEE, 2024.
- [9] A. S. Raghavendra, P. K. Suresh and M. Anitha, "Speech to Sign Language Translation for Indian Languages," IEEE Conference, 2023.
- [10] Z. Zhou, W. Li and Y. Xu, "Multi-Semantic Discriminative Feature Learning for Sign Gesture Recognition Using Hybrid Deep Neural Architecture," IEEE Access, 2023.
- [11] W. Stokoe, D. Casterline, and C. Croneberg, A Dictionary of American Sign Language on Linguistic Principles. Gallaudet University Press, 1965.
- [12] R. Bowden et al., "Software Glove for Sign Language Interpretation," IEEE Trans. Syst., Man, Cybern., vol. 34, no. 3, pp. 1555–1567, 2004.
- [13] S. Koller, O. Zargaran, H. Ney, and R. Bowden, "Deep Sign: Robust Statistical Sign Language Recognition via Hybrid CNN-HMMs," in Proc. BMVC, 2016.
- [14] S. Stoll, N. Camgoz, S. Hadfield, and R. Bowden, "Sign Language Production Using Neural Machine Translation and GANs," in Proc. BMVC, 2018.
- [15] B. Bird and E. Klein, Natural Language Processing with Python. O'Reilly Media, 2009.
- [16] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed. Pearson, 2022.
- [17] Google LLC, "Google Cloud Speech-to-Text Documentation," [Online]. Available: <https://cloud.google.com/speech-to-text>. [Accessed: Mar. 2025].
- [18] Django Software Foundation, "Django Web Framework Documentation," [Online]. Available: <https://docs.djangoproject.com>. [Accessed: Mar. 2025].
- [19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, pp. 4171–4186, 2019.
- [20] N. C. Camgoz, S. Hadfield, O. Koller, H. Ney, and R. Bowden, "Neural Sign Language Translation," in Proc. IEEE CVPR, 2018.