WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Mozhi: An AI-Based Indian Sign Language Text-to-Sign Converter

Mrs. M Manimegalai[#1],C Ganasri[#2],N S Girivaasan[#2],D Gobika[#2],I Gokul[#2]

[#1]Assistant Professor,Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India
[#2]Student, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, India.

A B S T R A C T :

Communication between deaf and hearing people can be difficult when they do not know the same language. Indian Sign Language (ISL) helps many deaf and hard-of-hearing people communicate, but people who do not know ISL may find it hard to understand signs. This paper presents Mozhi, a web-based Indian Sign Language text-to-sign converter designed to make communication easier. The system takes text entered by a user and presents a matching sign through a visual signing area. It also keeps a history of previous translations so users can review them later. The project focuses on a simple interface, clear results, and easy access to past translations. The proposed design includes a text input area, a translation action, a visual sign display, and a history section. The backend is planned to connect text processing with an ISL sign library or sign animation model, while the frontend remains consistent with the existing Mozhi design. This paper explains the problem, system design, main modules, workflow, testing plan, benefits, limitations, and future improvements. Mozhi is intended as an assistive communication tool and not as a replacement for interpreters or human communication.

Keywords: Indian Sign Language, Sign Language Converter, Accessibility, Sign-to-Text, Computer Vision, Web Application

1. Introduction

Communication is an important part of education, work, healthcare, and daily life. People who use Indian Sign Language may face barriers when communicating with people who do not understand signs. Written text can help in some situations, but it does not always provide the direct visual support needed for sign language communication. A simple digital tool can help users see how a written phrase may be expressed through signs.

Mozhi is a web application project that aims to connect text with visual Indian Sign Language output. The name Mozhi means language. The project is designed around the idea that communication should be more inclusive and easy to access. A user enters a word or sentence, starts translation, and views the sign output in a visual area. Each translation can be saved in a nearby history panel, allowing users to return to earlier results.

The project gives special attention to the current frontend. The existing pages, colours, navigation, and layout are intended to remain unchanged while backend features are connected. This helps keep the user experience familiar and avoids unnecessary redesign. The main goal is to build a useful and understandable prototype that can be improved as a larger ISL sign dataset becomes available.

Nomenclature

- ISL – Indian Sign Language
- AI – Artificial Intelligence
- UI – User Interface
- API – Application Programming Interface
- ML – Machine Learning
- TTS – Text-to-Speech
- Input Text – Text entered by the user for translation
- Sign Animation – Visual representation of sign actions
- Translation History – Stored record of previous translations

1.1. 1.1 Problem statement

Many people do not know Indian Sign Language, and this can make everyday conversations difficult. General text-to-speech tools do not show sign movements. Some sign language systems focus on other sign languages or only recognize signs from a camera. There is a need for a simple web-based tool that accepts text, displays corresponding ISL signs, and stores previous translations for quick review.

1.2. 1.2 Objectives

The main objectives of Mozhi are to: (i) accept text from the user; (ii) map text to suitable ISL signs; (iii) show the sign output visually, preferably through a signing avatar, animation, or sign clips; (iv) save each translation in a history section; (v) allow users to search or revisit saved translations; and (vi) provide an accessible and easy-to-use interface.

2. Related Work

Sign language technology includes sign recognition, sign generation, fingerspelling, and translation systems. Sign recognition usually takes a video or image of a person signing and predicts a label or sentence. Sign generation works in the opposite direction by presenting signs for text or speech. These tasks are related, but they need different data and processing methods.

Research and public tools show that computer vision and machine learning can support sign language applications. However, sign languages are not the same across countries. American Sign Language, British Sign Language, and Indian Sign Language use different signs and grammar. A system intended for Indian users should therefore use ISL-specific examples and should be checked by people who understand ISL.

A further challenge is that a sentence cannot always be converted by replacing each English word with one sign. Sign languages have their own grammar, facial expressions, and movement patterns. For this reason, Mozhi should be developed as a practical learning and communication aid, with clear limits stated when a complete sentence cannot be represented accurately.

3. Proposed System

Mozhi is planned as a web application with a frontend and a backend. The frontend provides the text entry field, translation button, visual output area, and translation history. The backend receives the text, cleans and processes it, finds suitable ISL sign data, and sends the result back to the interface. The history module stores the original text, translated sign sequence, date and time, and any available status information.

1.3. 3.1 System architecture

The system follows a simple flow: User Input → Text Processing → ISL Sign Mapping → Visual Sign Output → History Storage. The user enters a word or sentence. The backend normalizes the input and splits it into meaningful units. The mapping module searches the available ISL sign library. The frontend then plays the matching sign animation, avatar sequence, or video clip. Finally, the translation is added to history so it can be opened again.

The sign mapping source may be a prepared library of verified ISL clips or animations. If an avatar model is used, the system must map each sign to suitable movement data. If a word or phrase is not available, the system should show a clear message rather than presenting an incorrect sign. This fallback behaviour is important for user trust.

1.4. 3.2 Main modules

Text Input Module: collects the word or sentence typed by the user and checks that the field is not empty.

Text Processing Module: removes unnecessary spaces, handles basic punctuation, and prepares the text for sign mapping. More advanced versions may include sentence splitting and grammar rules.

ISL Mapping Module: searches the sign library and builds the sign sequence. It should use ISL-specific data and keep a record of missing words.

Visual Output Module: displays the selected sign through an animation, avatar, or video clip. Playback controls such as play, pause, and replay can make the result easier to follow.

History Module: stores each translation and shows it in the history panel. A user should be able to select an earlier entry and view its text and sign output again.

User Interface Module: connects the frontend to backend responses without changing the existing Mozhi page design. It should display loading, success, and error messages in a clear way.

1.5. 3.3 Suggested technologies

The frontend can use the existing HTML, CSS, and JavaScript or React implementation. A Python backend, for example using Flask or FastAPI, can handle text processing and API requests. A database such as SQLite can be used for a small prototype, while a larger deployment may use PostgreSQL or another supported database. Sign clips can be stored as local media files or served from a structured sign library. The final choice should depend on the project environment and the available ISL data.

4. System Workflow

Step 1: The user opens the Mozhi website and enters text in the translation field.

Step 2: The frontend checks that text has been entered and sends it to the backend.

Step 3: The backend normalizes the input and separates it into words or supported phrases.

Step 4: The ISL mapping module searches the available sign library for each item.
Step 5: The backend prepares an ordered sequence of sign clips or animation instructions.
Step 6: The frontend displays the visual signs and allows the user to replay them.
Step 7: The system saves the input, output reference, and timestamp in translation history.
Step 8: The user can select a history item to view the previous translation again.

If some words are not available, the system should identify them and explain that the translation is incomplete. It should not silently guess a sign. This makes the output more transparent and gives the project a clear path for adding new sign data.

5. Implementation Design

1.6. 5.1 Frontend

The Mozhi frontend uses a modern and accessible design, with a purple and teal colour theme and clear navigation. The current page structure includes Home, Open Camera, History, Products, Blog, About, and Contact pages. For the text-to-sign feature, the important elements are the text input area, translation action, sign display panel, and history panel. The existing visual layout should be preserved while API connections are added.

The frontend sends the user's text to a backend endpoint and waits for a response. During processing, it can show a loading indicator. If the request succeeds, the sign output is displayed and the new translation appears in history. If an error occurs, the page should show a short message and allow the user to try again.

1.7. 5.2 Backend and data storage

The backend acts as the connection between user input and sign content. A translation record may contain an identifier, original text, list of matched signs, media paths, creation time, and translation status. The history page can request recent records from the backend and display them in order. Search and date filters can be added to help users find older entries.

For a prototype, a database table can store translation records, while a separate sign table stores the word or phrase, sign label, media path, and language notes. The system should avoid storing personal information that is not needed. If history can be deleted by the user, that option should be clear and reliable.

1.8. 5.3 Sign presentation

A sign can be presented using a short video clip, a sequence of prepared animations, or a 3D avatar. A real-looking avatar is useful, but it requires reliable movement data and careful testing. A clip-based system may be easier to build first if a verified ISL dataset is available. The selected method should preserve hand shape, movement, position, and facial cues where these are part of the sign.

English text and ISL do not always follow the same word order. A basic prototype may map known words one by one, but this should be described as a limited text-to-sign system rather than a complete language translator. Future versions can add ISL grammar rules, phrase-level signs, and expert-reviewed sentence patterns.

7. Testing and Evaluation

Testing should check whether the main features work correctly and whether the displayed signs match the selected ISL data. The prototype can be tested using a list of common words and short phrases that exist in the sign library. Each test should record the input, expected behaviour, actual behaviour, and result. No accuracy score should be reported until tests have been carried out and the results have been measured.

1.9. 7.1 Functional tests

Input validation: an empty input should show a helpful message. Known sign: a supported word should return the correct sign media. Unknown word: an unsupported word should be clearly marked. History: a successful translation should appear in the history list. History replay: selecting an older entry should reopen its saved sign output. Error handling: if the backend is not available, the page should show an error without breaking the layout.

1.10. 7.2 Evaluation measures

Useful measures include the percentage of supported test items that return the expected sign, response time, number of missing words, successful history saves, and feedback from users who know ISL. A sign language expert or trained ISL user should review the signs before the system is used for important communication. User feedback can also help identify confusing controls or signs that are difficult to understand.

8. Benefits and Applications

Mozhi can help beginners explore ISL signs, support basic communication practice, and provide a visual reference for common words. It may be useful in learning environments, public service desks, shops, workplaces, and at home. The history feature helps users repeat earlier translations without typing the same text again. A web-based design also makes the prototype easier to access on devices with a browser.

The project gives students practical experience with frontend development, backend APIs, database design, media handling, and accessible user interface design. It also demonstrates how software can be built around a real communication need.

9. Limitations

The quality of the output depends on the amount and quality of the ISL sign data. If a sign is missing, the system cannot provide a reliable visual result for that word. Direct word-by-word mapping may not express the grammar or meaning of a full ISL sentence. Avatar movement can also look unnatural if the animation data is incomplete. The project should therefore be presented as a support tool and not as a certified replacement for a human interpreter.

The current paper describes the proposed design and testing plan. Actual performance, user acceptance, and translation accuracy must be reported only after the implementation has been tested. Privacy and consent should also be considered if user history or camera-based features are added.

10. Future Enhancements

Future work can include a larger ISL sign library, expert-reviewed phrase translations, improved sentence processing, and a smoother signing avatar. Speech-to-text could allow spoken input, while camera-based sign recognition could support the reverse direction. A mobile-friendly version, offline access to common signs, playback speed controls, and user feedback options may also improve usability. Collaboration with deaf users and ISL experts should guide the next versions.

11. Conclusion

Mozhi is a web-based project designed to make Indian Sign Language more accessible through text input, visual sign output, and translation history. The proposed system connects a simple frontend with backend text processing, ISL sign mapping, and storage of previous translations. Keeping the existing frontend unchanged helps preserve the design already created for the project. The main challenge is to provide accurate ISL signs and represent sentence meaning correctly. By using verified sign data, clear messages for unsupported words, and careful testing with ISL users, Mozhi can grow into a more useful communication and learning aid.

Acknowledgements

The author thanks the faculty and peers who provide guidance and feedback during the development of the Mozhi project.

References

- World Health Organization. (2021). World report on hearing. World Health Organization.
- Ministry of Social Justice and Empowerment, Government of India. Indian Sign Language Research and Training Centre (ISLRTC). Official website and learning resources.
- Indian Sign Language Research and Training Centre. Indian Sign Language dictionary and related learning resources.
- Bragg, D., Koller, O., Bellard, M., Berke, L., Boudreault, P., Braffort, A., et al. (2019). Sign language recognition, generation, and translation: An interdisciplinary perspective. Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility.
- World Wide Web Consortium (W3C). Web Content Accessibility Guidelines (WCAG) 2.2. W3C Recommendation