



International Conference on Education, Psychology and Humanities

Hosted Online from Moscow, Russia

Date: 28th June, 2026

Website: <https://econferencia.com>

SYSTEM ARCHITECTURE AND ACCESSIBLE INTERFACE DESIGN FOR REAL-TIME UZBEK SIGN LANGUAGE TRANSLATION

Jasurbek Ergashev¹,

Munira Rakhmatova²,

Mukhtor Eshimov³

Department of Computer Science, School of Engineering,
Central Asian University, Tashkent, Uzbekistan

Abstract

Real-time sign-language translation systems are usually evaluated on recognition accuracy alone, but accuracy is only useful if the surrounding system actually reaches its intended users. This paper describes the system architecture and accessible interface design behind our Uzbek Sign Language (UzSL) translation prototype, developed as part of a broader sign-language translation project [1]. We separate video capture from model inference across a client-server architecture. Frames stream from an in-browser client to a FastAPI backend over WebSockets [4], and the recognition model is served through ONNX Runtime or PyTorch and deployed in a Docker container [5, 6] for portability across environments. The web interface, built with Next.js, Tailwind CSS, and Radix UI [7], was evaluated against the Web Content Accessibility Guidelines (WCAG) 2.1 [2], including screen-reader compatibility, keyboard-only navigation, and colour-contrast requirements. We report a series of refinements made in response to field observations during testing, including nighttime-recording support for low-light conditions, contextual thresholds to reduce confusion between visually similar signs, a caching layer that cut speech-synthesis latency by roughly 40% [3], and interface adjustments such as larger tap targets for younger and older users. We present these architectural and accessibility decisions as a practical



International Conference on Education, Psychology and Humanities

Hosted Online from Moscow, Russia

Date: 28th June, 2026

Website: <https://econferencia.com>

reference for deploying real-time recognition systems for low-resource sign languages, independent of the specific recognition model used.

Keywords: Uzbek Sign Language; system architecture; accessibility; WCAG; client-server design; real-time systems; assistive technology; human-computer interaction.

1. INTRODUCTION

Most sign-language recognition research reports model accuracy without much detail on how a model actually becomes a usable, accessible system for the people it is meant to serve. That gap matters for low-resource languages like Uzbek Sign Language (UzSL), where deployment constraints, limited infrastructure, a small developer base, and no existing commercial tooling, shape what kind of system is realistic to build and maintain [1]. This paper sets the model comparison aside. Instead, it documents the architecture and interface decisions behind our UzSL translation prototype: how the system is structured across client and server, and how the interface was designed to meet accessibility standards rather than added as an afterthought.

We treat these as engineering contributions in their own right. A recognition model is only as useful as the pipeline that delivers its output to a person in real time, and a translation tool that is not screen-reader compatible or keyboard-navigable excludes exactly the range of users it is meant to include. We describe the reasoning behind our client-server split, our choice of serving and deployment tools, the accessibility requirements we built and tested against, and a set of refinements that came directly out of testing with real users.



International Conference on Education, Psychology and Humanities

Hosted Online from Moscow, Russia

Date: 28th June, 2026

Website: <https://econferencia.com>

The rest of this paper covers system architecture (Section 2), interface design for accessibility (Section 3), refinements made from field observations (Section 4), and a discussion of the tradeoffs involved (Section 5).

2. SYSTEM ARCHITECTURE

The prototype uses a client-server architecture, chosen specifically to separate the concerns of capture and inference. The client runs entirely in a standard web browser, requiring no installation: it captures live video from the user's webcam and streams frames asynchronously to the server over WebSockets, a protocol suited to the continuous, low-latency exchange real-time gesture recognition requires [4]. This separation keeps the client lightweight while the computationally heavier recognition work happens server-side.

On the server, the recognition model runs through either ONNX Runtime, for optimized inference once the model has been converted, or PyTorch directly, depending on deployment needs [6]. The server itself runs inside a Docker container [5]. That keeps the environment consistent across development, testing, and production, and makes cloud deployment straightforward. It also handles multiple inference sessions in parallel through asynchronous processing queues, so it stays responsive to new requests even when several users are connected at once.

Component	Role
Web client (Next.js)	In-browser video capture; no installation required
WebSocket connection	Asynchronous, low-latency frame streaming to the server
FastAPI backend	Asynchronous request handling and pre/post-processing
Recognition model (ONNX Runtime / PyTorch)	Server-side inference on incoming frames
Docker container	Consistent, portable deployment across environments



International Conference on Education, Psychology and Humanities

Hosted Online from Moscow, Russia

Date: 28th June, 2026

Website: <https://econferencia.com>

3. ACCESSIBLE INTERFACE DESIGN

The front end was built with Next.js, styled with Tailwind CSS, and uses Radix UI components for accessible interaction primitives [7]. Beyond its core function, live webcam preview, real-time gesture-to-text output, and speech playback of translations, we treated accessibility as a design requirement rather than a later addition.

We tested the interface for compatibility with screen readers and for full keyboard-only navigation, and checked the colour palette against contrast-ratio guidelines for colour-blind accessibility, following the Web Content Accessibility Guidelines (WCAG) 2.1 [2]. Interaction logging, collected through the interface itself, helped surface friction points that were not obvious from design review alone: teacher feedback on the companion mobile app, for example, pointed to tap targets that were too small for younger and older users, which led us to enlarge interactive elements across the interface.

Translation output is also logged in a visible history so users can review the sequence of signs recognized rather than seeing only the latest prediction, a change made after early users asked to check the system's interpretation over a longer exchange rather than one gesture at a time.

4. FIELD-INFORMED REFINEMENTS

Several architectural and interface decisions came directly out of testing rather than upfront design. Recognition confidence dropped in low-light conditions, which we addressed by expanding the training data with nighttime recordings and integrating a hand-segmentation module to isolate hand regions more reliably. Visually similar signs, such as the UzSL signs for “thank you” and “sorry,” were often confused by the model; we reduced this by adding more training examples



International Conference on Education, Psychology and Humanities

Hosted Online from Moscow, Russia

Date: 28th June, 2026

Website: <https://econferencia.com>

for the affected signs and introducing a contextual time-delay threshold before committing to a prediction.

On the server side, repeated text-to-speech requests for common phrases sent to the Google Cloud Text-to-Speech API introduced noticeable delay [3]. Adding a caching layer for frequently repeated phrases cut average synthesis latency by roughly 40%, which made the voice-feedback feature feel considerably more responsive in practice.

5. DISCUSSION

Treating architecture and accessibility as first-class design concerns, rather than implementation details handled after a model is trained, changes what “finished” looks like for a system like this. Strong recognition accuracy alone does not make a usable communication tool: latency, interface clarity, and compliance with accessibility standards all determine whether the system is something a deaf or hard-of-hearing user would actually choose to rely on.

The client-server split has a real cost. It introduces a dependency on network connectivity and server-side compute that an offline, on-device approach would avoid. That tradeoff suits a research prototype tested mostly indoors on a stable connection, but it is a genuine limitation for the low-connectivity environments this project was partly motivated by in the first place. An edge-deployment path, running a lighter model directly on-device, is a natural complement to the server-based approach described here, though it comes with its own accuracy-versus-portability tradeoffs that would need separate evaluation.

Our accessibility testing was also limited in scope: WCAG 2.1 compliance and screen-reader/keyboard-navigation testing establish a baseline, but they were not validated directly with blind, low-vision, or motor-impaired users, which is a meaningfully different kind of evidence than internal compliance testing.



International Conference on Education, Psychology and Humanities

Hosted Online from Moscow, Russia

Date: 28th June, 2026

Website: <https://econferencia.com>

6. LIMITATIONS AND FUTURE WORK

Accessibility testing followed WCAG 2.1 guidelines but was not validated with users who have visual, motor, or other disabilities the guidelines are meant to serve. Server-side inference introduces a connectivity dependency not present in an on-device approach. The refinements described here - low-light handling, sign-confusion mitigation, TTS caching - were evaluated informally during testing rather than through controlled before/after comparison, and parallel-session handling under heavy concurrent load was not benchmarked. Future work should prioritize a formal accessibility evaluation with assistive-technology users, benchmarking of concurrent-session throughput under load, and a systematic comparison of server-based versus edge-inference latency and accuracy tradeoffs.

7. CONCLUSION

This paper set model accuracy aside on purpose, to focus on the architecture and interface decisions that determine whether a sign-language recognition system is actually usable and accessible in practice. Separating capture from inference, building the interface around WCAG 2.1 requirements from the start, and refining both based on real testing produced something more than a recognition demo. It is a starting point, not an endpoint. Future work should evaluate accessibility directly with assistive-technology users and more rigorously benchmark the tradeoffs between server-based and edge deployment for low-resource-language contexts.



International Conference on Education, Psychology and Humanities

Hosted Online from Moscow, Russia

Date: 28th June, 2026

Website: <https://econferencia.com>

REFERENCES

- [1] Ergashev, J., Eshimov, M., & Rakhmatova, M. (2025). Sign Language Translation Model. B.S. thesis, Central Asian University, Tashkent, Uzbekistan.
- [2] World Wide Web Consortium (W3C). (2018). Web Content Accessibility Guidelines (WCAG) 2.1. <https://www.w3.org/TR/WCAG21/>
- [3] Google Cloud. Text-to-Speech documentation. Retrieved 2025, from <https://cloud.google.com/text-to-speech>
- [4] FastAPI. Documentation. Retrieved 2025, from <https://fastapi.tiangolo.com>
- [5] Docker Inc. Docker documentation. Retrieved 2025, from <https://docs.docker.com>
- [6] ONNX Runtime. Documentation. Retrieved 2025, from <https://onnxruntime.ai>
- [7] Vercel. Next.js documentation. Retrieved 2025, from <https://nextjs.org/docs>