iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** XI **Month of publication:** November 2026

DOI: https://doi.org/10.22214/ijraset.2026.85697

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue XI Nov 2026- Available at www.ijraset.com

SignMeet AI: Real-Time Sign Language Translation for Inclusive Video Communication

Deepika Sarwate[1] Om Gore[2] Anam Inamdar[3]
Master of Computer Applications (MCA), JSPM's Narhe Technical Campus, Savitribai Phule Pune University, Pune, India

Abstract: ***This paper reviews recent work on webcam-based sign language recognition and proposes SignMeet AI, a browser-based meeting prototype aimed at some of the gaps in that work. Calls run peer to peer over Web Real-Time Communication (WebRTC) and PeerJS. MediaPipe Hands extracts 21 hand landmarks from each webcam frame, a machine-learning (ML) classifier maps them to American Sign Language (ASL) gestures, and a temporal stability filter (a confidence threshold, a five-frame consistency check and debouncing) keeps flickering predictions out of the output. Spoken words are transcribed in parallel, and Django Channels merges both streams into one timestamped transcript for everyone in the room. A meeting starts from a room ID, with no account needed. In sample tests, the stability filter cut unwanted predictions from 18 to 6 and repeated characters from 24 to 8 per session. Overall recognition accuracy was 87%, and text appeared after about 320 ms for signs and 342 ms for speech. These numbers come from a small prototype evaluation and should be read as indicative. The vocabulary is limited to static ASL gestures. Continuous signing, Indian Sign Language and a user study with signers remain future work.***
Keywords: Sign Language Recognition, MediaPipe, WebRTC, Video Conferencing, Speech-to-text, Stability Filtering, Django Channels, Accessibility, Real-time Translation.

I. INTRODUCTION

A. Background and Motivation

Video calls now carry a good share of everyday life: lectures, job interviews, clinic appointments, team meetings. The shift sped up during the COVID-19 pandemic and mostly stayed. For many people a call is no longer a convenience but the normal way to study and work.

Most conferencing tools, however, are built for speech. They assume everyone can hear the other participants and answer aloud. According to the World Health Organization (WHO) [1], over 1.5 billion people live with hearing loss of some degree. Roughly 430 million have a loss serious enough to be disabling, and the total is projected to reach nearly 2.5 billion by 2050. A large part of this community uses a sign language as its first or preferred language. For them, a call that supports only speech means relying on other people or being left out.

The usual workarounds only go part of the way. An interpreter makes a meeting accessible, but interpreters cost money, must be booked ahead, and are not available everywhere or at every hour. Captions and typed chat let a deaf participant follow what is said, yet they do nothing for someone who wants to answer in sign. Typing is slower than signing and drops most of the tone and expression. One side takes part fully while the other takes part through a workaround.

Sign languages such as Indian Sign Language (ISL) and American Sign Language (ASL) are complete natural languages with their own grammar and vocabulary. Meaning sits in the handshape, but also in movement, location and orientation, and often in facial expression and posture. That is why sign recognition is harder than classifying a fixed set of hand shapes, and why research has drifted from single images toward modeling motion over time. A good tool should treat signing as a language in its own right, not as a stand-in for text.

Several technologies have matured enough to make a practical tool plausible. MediaPipe can extract hand landmarks, the positions of the fingers and joints, from an ordinary webcam in real time, even inside a browser [2]. Machine-learning models can learn to classify those landmark patterns into signs. Speech recognition turns spoken words into text almost instantly, and WebRTC moves audio and video directly between browsers with little delay [3]. Each piece works on its own. Putting them into one usable meeting environment is the part that is still thin.

Running recognition in the browser was a deliberate choice. Frames are processed on the user's own device, so no round trip to a server is needed before a result appears, less personal video leaves the machine, and hosting costs less. A peer-to-peer call serves the same ends for audio and video.

©IJRASET: All Rights are Reserved

Three problems remain in this area:

- Unstable output. Recognition that works in a controlled demo wobbles under real lighting, hand angles and signing speeds. Raw frame-by-frame predictions flicker and repeat, which makes the text hard to read.
- Weak integration. Sign translation usually lives in a separate application or as an add-on to a conferencing platform, and many tools handle only one direction.
- Heavy models. The most accurate architectures can be too demanding to run in a browser during a live call.

SignMeet AI, the system proposed in this paper, responds to each of these. A stability filter steadies the output. One meeting environment holds the video call, sign recognition, speech transcription and a shared transcript. A light landmark-based pipeline runs on the user's own device. Signing and speaking participants can then follow one conversation without a separate interpreter.

B. Statement of the Problem

Deaf and hard-of-hearing people meet a persistent barrier in mainstream video conferencing. Platforms treat speech as the main channel and give little support to sign language, which is the natural language of many deaf users. Signers end up relying on an interpreter, on typed chat, or on the patience of the other participants, and none of these gives them the ease that hearing users have.

The literature shows real progress on separate pieces of the problem: webcam hand detection, landmark extraction, deep-learning classification, even continuous recognition. Three difficulties are still open. Systems that look good in a controlled demonstration become unstable when lighting, hand orientation and signing speed change, and raw frame-level output flickers and repeats. Many studies stop at isolated gestures, while conversation needs continuous, context-aware interpretation. And accurate models are often too heavy for a browser during a live call. The difficulty is not only technical. When recognized words arrive late, arrive twice or are wrong, participants spend the meeting correcting and guessing. A signer may give up, and hearing participants may stop trusting the tool. Usability therefore depends on stability, speed and how natural the output feels, not on accuracy alone. Integration is the second gap. Many tools treat sign translation as a separate application or an add-on over a conferencing platform. Others convert signs into text or speech but offer no equally natural channel in the other direction. Few bring video, sign recognition, speech recognition and a shared transcript together in one environment. The problem addressed here is the lack of a practical, low-latency, browser-based video-conferencing system that recognizes sign language in real time, transcribes speech at the same time, and shows both in one synchronized transcript for all participants, without gloves, special sensors or a human interpreter.

C. Objectives of the Research

The main aim is a real-time, AI-powered framework that makes video conferencing more accessible to deaf and hard-of-hearing users by combining sign recognition, speech transcription and peer-to-peer video. The study sets eight specific objectives:

- Review recent research on sign language recognition, hand-landmark detection, continuous recognition and bidirectional sign and speech translation, and identify the gaps that remain.
- Design a peer-to-peer video-conferencing architecture with WebRTC and PeerJS that supports low-latency audio and video and simple room-based sessions.
- Implement in-browser hand detection and landmark extraction with MediaPipe Hands, so that recognition works from an ordinary webcam.
- Develop a machine-learning classifier that maps hand-landmark patterns to sign gestures.
- Introduce a temporal stability filter that reduces false and repeated predictions and makes the recognized output readable.
- Build a parallel speech-to-text pipeline whose text events travel in real time through Django Channels.
- Merge sign recognition output and speech transcription into one synchronized transcript visible to every participant.
- Evaluate the prototype for recognition accuracy, response time and usability, and note directions for improvement.

D. Research Questions and Hypotheses

The study addresses three research questions. RQ1: Does a temporal stability filter reduce false and repeated predictions compared with raw frame-by-frame classification? RQ2: Can a browser-based pipeline of MediaPipe Hands and a lightweight classifier recognize signs fast enough for a live call? RQ3: Does a unified, synchronized transcript help signing and speaking participants follow the same conversation more easily than separate or one-directional tools? Each is paired with a hypothesis:

H1: The stability filter significantly reduces false and repeated sign predictions.

H2: A browser-based pipeline of MediaPipe Hands and a lightweight classifier recognizes signs fast enough to support real-time communication.

H3: A shared transcript of sign and speech output makes the conversation easier to follow.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Each hypothesis is formulated as a testable statement with a null counterpart. H0-1 states that the stability filter makes no significant difference to the number of false or repeated predictions. H0-2 states that a browser-based pipeline of this kind cannot recognize signs fast enough for real-time communication. H0-3 states that a shared transcript brings no improvement in how easily participants follow the conversation. H1 is examined by counting false and repeated predictions with and without the filter on the same gestures. H2 is examined by timing how long landmark detection and prediction take, against what a live conversation needs. H3 would be examined through feedback from participants who use the shared transcript, compared with their experience of separate tools. Section V states which of these the current prototype can support.

As this study centers on a prototype and small sample tests, the hypotheses are examined through the sample tests in Section V and not through a statistical significance test. Section V states which of them the current prototype can support.

E. Significance of the Study

- *For deaf and hard-of-hearing users:* equal participation. A deaf student or employee who cannot join video calls comfortably is at a real disadvantage. A system in which a signer signs, hearing participants speak, and both read the same transcript reduces that gap. A student could follow an online lecture, a signing employee could join a team meeting or interview, and in healthcare or public services even a modest tool that reduces misunderstanding matters. In each case the gain is independence: nobody has to speak on the user's behalf.
- *For institutions and small organizations:* a lower cost and scheduling burden than arranging an interpreter for every meeting. This is especially relevant in settings where interpreters are hard to arrange.
- *Technically:* the study shows how existing components can be combined in one working design. Browser-side recognition keeps the heavy processing on the user's device. Landmarks give the classifier a compact input that ordinary laptops can handle. The stability filter targets a very practical problem, the flicker and repetition that make many recognition demos unusable in conversation. Room-based sessions keep the entry barrier low.
- *For research:* a structured review of recent work, a set of specific gaps (isolated rather than continuous recognition, one-directional translation, heavy models, weak conferencing integration), and an architecture that others can test and extend. Work on larger vocabularies, continuous recognition, multiple sign languages and context-aware translation can build directly on it. The study also treats accessibility as a starting point of the design, not a feature added at the end.

F. Scope and Delimitations

The scope is deliberately practical. The prototype works with a defined vocabulary of static ASL signs captured by a standard webcam and supports small room-based meetings. It does not model the full grammar of any sign language and does not generate sign video from text. Those larger goals are discussed as future scope in Section VII.

The study is a prototype study. It reports sample tests, not a controlled benchmark with a fixed vocabulary, repeated trials and a larger group of signers, and it includes no user study with signers. These delimitations keep the questions narrow enough to be answered by the prototype and its sample tests, and they mark clearly what the reported figures can and cannot show.

II. REVIEW OF LITERATURE

Researchers in computer vision and deep learning have spent the last few years trying to automate sign language recognition and translation, mainly to ease conversation between deaf or hard-of-hearing people and hearing users. The traditional options, interpreters and typed text, slow a conversation down and strip out its natural flow. This section reviews recent work in five areas that bear on the objectives in Section I: webcam-based recognition, MediaPipe and deep-learning pipelines, recognition inside video conferencing, two-way sign and speech communication, and continuous and multimodal systems. It closes with the research gap.

A. Webcam-Based Real-Time Recognition

Sangeetha Priya et al. [4] built a translation system that works from a live webcam. It extracts hand landmarks, identifies the sign being made and outputs text. They value landmarks because these capture the spatial configuration of the hand without special sensing hardware, and they list what limits real-world use: processing latency, ease of use, lighting, hand orientation, vocabulary size and dynamic signs. SignMeet AI shares the webcam-only setup and most of these concerns. Other recent systems follow the same webcam pattern: Aabid [5] describes a real-time sign language translator for specially abled users, and Lokhande et al. [6] present another real-time sign language translator.

Sonare et al. [7] studied video-based sign translation with machine learning. Their work shows the benefit of analyzing sequences of frames instead of single images, since signing unfolds over time. The price is extra computation and temporal processing, which has to be weighed carefully in a low-latency call.

Moryossef et al. [8] used human pose estimation to detect signing in real time. Pose gives a structured description of the body and depends less on raw pixels. That supports a choice made in SignMeet AI: the classifier receives hand landmarks, not images.

B. MediaPipe and Deep-Learning Pipelines

Natarajan et al. [9] proposed an end-to-end framework covering recognition, translation and sign-video generation. It extracts pose information with MediaPipe and passes it to a hybrid convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM) model. Their results suggest that joining spatial feature extraction with temporal sequence modeling recognizes signs better than treating each gesture as a separate image. Generating sign video from text also makes the work relevant to two-way systems.

SignMeet AI relies on MediaPipe in the same way but has a narrower aim: practical real-time communication in the browser, without a video-generation stage. Using landmark extraction as a light front end leaves the browser's resources free for classification and synchronization.

C. Recognition Inside Video Conferencing

Vishwekar et al. [10] proposed SMART SIGN, a live interpretation system for video conferencing that works with common platforms and produces written captions from recognized gestures. Their case is that sign recognition belongs in the meeting itself, so users need not switch to a separate translation application.

Mukherjee et al. [11] caption sign gestures during video meetings with a browser extension that turns ASL gestures into subtitles for the other participants. They target word-level ASL and apply deep learning to more than 2,000 word-level videos performed by over 100 signers. The work shows that recognition can join online meetings without gloves or three-dimensional sensors.

Both approaches mainly add translation on top of conferencing platforms that already exist. SignMeet AI tries something more integrated: a peer-to-peer meeting application in which video, translation and transcript synchronization are designed as connected parts of one system.

D. Two-Way Sign and Speech Communication

Many systems stop at sign-to-text or sign-to-speech. That helps a deaf user reach a hearing one, but it gives no equally natural path in the other direction.

Dey and Paul [12] combined Indian Sign Language recognition with real-time speech synthesis for video conferencing, which shows why sign recognition has to connect with ordinary spoken communication. In a linguistically diverse country like India, ISL can serve as an important accessibility layer for digital communication. Pradeep et al. [13] proposed an AI-powered bidirectional sign language translator for real-time mobile voice calls, treating communication as two-way translation and not just caption display. Nayak et al. [14] report a real-time sign language chat application with speech output, another pairing of gesture recognition with everyday speech. C. S. [15] likewise combines sign language translation with real-time audio transcription.

Taken together, these studies support a design principle for SignMeet AI: both streams should be available at the same time. The system pairs sign recognition with speech-to-text and puts both in a shared transcript, so hearing and signing participants follow the same conversation without a human interpreter.

E. Continuous Recognition and Transformer Models

The field has moved gradually from isolated gestures toward continuous recognition, where a system must pick out whole sequences of signs and read them in context. Said et al. [16] proposed an adaptive Transformer-based framework for continuous sign language recognition and translation. Transformers handle sequences well because they relate elements across long spans, and that matters in signing: the meaning of a sign is not always fixed by one hand shape, and movement, order and context change how it reads. R et al. [17] studied multilingual dynamic sign translation with Vision Transformers, and Shinde and Kannan [18] combined several neural architectures for sign video translation. Both cover the spatial and the temporal side of signing.

These approaches promise better accuracy and support for continuous signing, but they can demand a lot of computing power. In a browser-based call, model size, inference speed, memory and network latency become practical constraints. SignMeet AI therefore keeps its recognition pipeline light and treats continuous recognition as future work.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

F. End-to-End and Multimodal Systems

Work is also moving toward complete systems that link the stages of sign communication. Tantaroudas et al. [19], [20] describe an AI-driven extended reality (XR) framework that includes real-time sign interpretation and emotion or sentiment analysis, and Srivastava et al. [21] investigate bidirectional translation in the SignSpeak framework. Together with the work of Natarajan et al. [9], and with the gesture-aware neural networks that Waheed et al. [22] apply to real-time recognition on aerial robotic platforms, these studies show a shift from gesture recognition toward multimodal conversational systems. SignMeet AI follows that direction but stays with one practical scenario: a video meeting in which sign recognition and speech transcription meet in one interface.

G. Research Gap Summary

The reviewed work shows substantial progress in sign language recognition, yet four gaps remain. Table I sets each against the focus of the present research.

TABLE I
RESEARCH GAP SUMMARY

Aspect	Existing Research	Identified Gap	Present Research Focus
Recognition scope	Many systems recognize isolated gestures or words.	Conversation needs continuous, context-aware recognition.	A stability filter that makes frame-level output readable now; continuous recognition is planned (Section VII).
Communication direction	Several systems stop at sign-to-text.	No equally integrated path in the other direction.	Speech-to-text and sign recognition feed one shared transcript.
Model weight	The most accurate deep-learning models are heavy.	Too heavy for low-latency use in a browser.	Landmark input (63 values per hand) and a lightweight classifier in the browser.
Integration	Recognition is often an add-on to a conferencing tool.	Rarely built into the conferencing environment itself.	One application combining peer-to-peer video, in-browser recognition, speech-to-text and a shared transcript.

The contribution is therefore not sign recognition on its own. It is the combination of recognition and spoken-language transcription in a single, low-latency meeting environment: webcam-based recognition with MediaPipe landmarks and in-browser inference, speech-to-text, a shared transcript, peer-to-peer WebRTC video and Django Channels synchronization, with no need for gloves, sensors or a human interpreter. The review is selective and not a systematic survey, so the combination is new only with respect to the studies reviewed here.

H. Comparative Review and Implications for the Present Design

Table II summarizes how each reviewed study relates to SignMeet AI. The grouping is conceptual: webcam and landmark systems, in-meeting integrations, two-way systems and sequence-model approaches. The studies that the review draws on span all of these groups, and SignMeet AI borrows from the first two while leaving the sequence-model approaches as future work.

TABLE II
COMPARATIVE ANALYSIS OF EXISTING STUDIES

Study	Main approach	Key contribution	Relevance to SignMeet AI
Sangeetha Priya et al. [4]	Webcam + hand landmarks	Real-time computer-vision translation	Supports lightweight webcam recognition
Sonare et al. [7]	Video + ML	Video-based sign recognition	Supports video-based gesture analysis
Moryossef et al. [8]	Human pose estimation	Real-time pose-based sign detection	Supports landmark-based recognition

Study	Main approach	Key contribution	Relevance to SignMeet AI
Natarajan et al. [9]	MediaPipe + CNN/Bi-LSTM	End-to-end recognition and translation	Supports landmark and temporal modeling
Vishwekar et al. [10]	AI + virtual camera	Sign interpretation for video conferencing	Directly relevant to conferencing
Mukherjee et al. [11]	Deep learning + browser extension	Real-time sign captions in meetings	Supports browser-based meeting translation
Dey and Paul [12]	ISL + speech synthesis	Sign recognition for video conferencing	Relevant to Indian accessibility
Pradeep et al. [13]	Bidirectional AI	Sign translation for voice calls	Supports two-way communication
Nayak et al. [14]	Chat application + speech output	Real-time sign chat with speech	Pairs gesture recognition with speech
Said et al. [16]	Transformer	Continuous sign recognition	Supports future continuous-sign capability
R et al. [17]	Vision Transformer	Multilingual dynamic translation	Supports future multilingual expansion
Shinde and Kannan [18]	Hybrid deep learning	Sign video translation	Supports spatial and temporal modeling
Srivastava et al. [21]	Deep learning	Bidirectional sign translation	Supports unified two-way architecture
Tantaroudas et al. [19], [20]	AI + XR	Real-time sign interpretation	Points to future multimodal accessibility
Aabid [5]	Real-time translator	Sign translation for specially abled users	Supports real-time webcam translation
Lokhande et al. [6]	Real-time translator	Real-time sign language translation	Supports real-time recognition
C. S. [15]	AI + audio transcription	Sign translation with real-time audio transcription	Pairs sign recognition with transcription
Waheed et al. [22]	Gesture-aware neural network	Recognition on aerial robotic platforms	Shows wider deployment of real-time recognition

III. RESEARCH METHODOLOGY / RESEARCH DESIGN

A. Research Design

The study follows a modular prototype approach. Each module is built and tested on its own, then integrated into the full system, and the finished prototype is evaluated against the measures in Section III-E. The work proceeds in five stages:

- Literature review and gap identification
- Architecture design
- Module-wise implementation
- Integration of the modules
- Evaluation and analysis

For evaluation, the independent conditions were whether the stability filter was on or off and the category of gesture. The dependent measures were recognition accuracy, unwanted and repeated predictions, and delay.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

B. Proposed Approach

SignMeet AI is built from five small modules instead of one large system. Each answers a question raised by the objectives: how participants see and hear each other, how a hand shape becomes a letter, how flicker is kept out of the text, how everyone follows one shared conversation, and how a meeting starts with as little effort as possible. One idea runs through the design. The heavy work, video handling and gesture recognition, happens in each user's browser. The server stays light and only tracks who is in a room and what has been said.

A user creates or joins a room, and a video call is set up. From then on two streams run side by side: the camera feeds the sign-recognition pipeline and the microphone feeds a speech-to-text pipeline. Both end in the same shared transcript, which goes to every participant (Fig. 1).

The five modules answer the objectives as follows.

- Module 1, peer-to-peer video (Objective 2): direct media transmission with WebRTC and PeerJS gives real-time audio and video.
- Module 2, ASL detection (Objectives 3 and 4): a webcam frame becomes hand landmarks and then a classified gesture, using MediaPipe Hands and an ML classifier.
- Module 3, stability filtering (Objective 5): a confidence threshold, temporal smoothing and debouncing give stable gesture text.
- Module 4, shared transcript (Objectives 6 and 7): sign and speech events are merged and broadcast with Django Channels over WebSocket.
- Module 5, session management (Objective 2): room ID generation with lightweight session state lets people create and join meetings.

C. Data and Trust Model

The model keeps each kind of data where it is produced. Video frames are analyzed on the user's own device, so no frame is sent to a server for recognition. Only small text events leave the browser, and the server keeps only lightweight room state: the host, the participant list, peer IDs and the transcript synchronization state.

Participants trust one another through the room ID. There are no permanent accounts; each person receives a temporary identity on entering, such as Participant 1 or Participant 2.

Every transcript event carries four fields:

- Speaker ID: shows who produced the message, for example Participant 1.
- Input type: marks the message as sign or speech, so the interface can show a sign icon or a microphone icon.
- Text: the recognized or translated words.
- Timestamp: keeps messages in the correct order for everyone.

A video-conferencing prototype handles people's faces, hands and voices, so privacy matters even at this stage. Recognition runs on the user's device, and raw video is not uploaded for recognition. The Django server does not carry the raw media when a direct peer-to-peer connection works. Joining by room ID without a login suits a prototype but not real-world use: a production version would need authentication, protected rooms, expiring links and secure connections (Section VII). The study makes no claim about production-grade security.

D. Experimental Procedure

The evaluation follows a short, fixed procedure on sample sessions of the prototype:

1) Set up a room and a call between participants with a standard webcam and microphone.
2) Perform gestures from the three categories (static alphabet gestures, common gestures, gestures with similar hand shapes) and record how often the correct sign is identified.
3) Run the same gestures with the stability filter off and on, and count unwanted predictions and repeated characters per session.
4) Time how long text takes to appear for signs and for speech, and check that transcript events reach every participant.
5) Compare the observations with the hypotheses and note what the sample cannot show.

Because the sample is small, the procedure should be repeated on more sessions and signers before the figures are treated as stable (Section V-G).

E. Evaluation Metrics

Once the system is running, it is evaluated with the measures in Table III.

©IJRASET: All Rights are Reserved

TABLE III
EVALUATION METRICS

Metric	What it tells us	Related module
Gesture recognition accuracy	How often the correct sign is identified	ASL detection
Classification confidence	How sure the model is about its predictions	ASL detection
False-positive rate	How often a wrong or accidental letter is accepted	Stability filter
Translation latency	Delay from making a sign to seeing the text	ASL detection, transcript
Transcript synchronization delay	How quickly all participants see the same message	Shared transcript
WebRTC connection performance	Connection setup time and stream quality	Video conferencing

F. Research Workflow

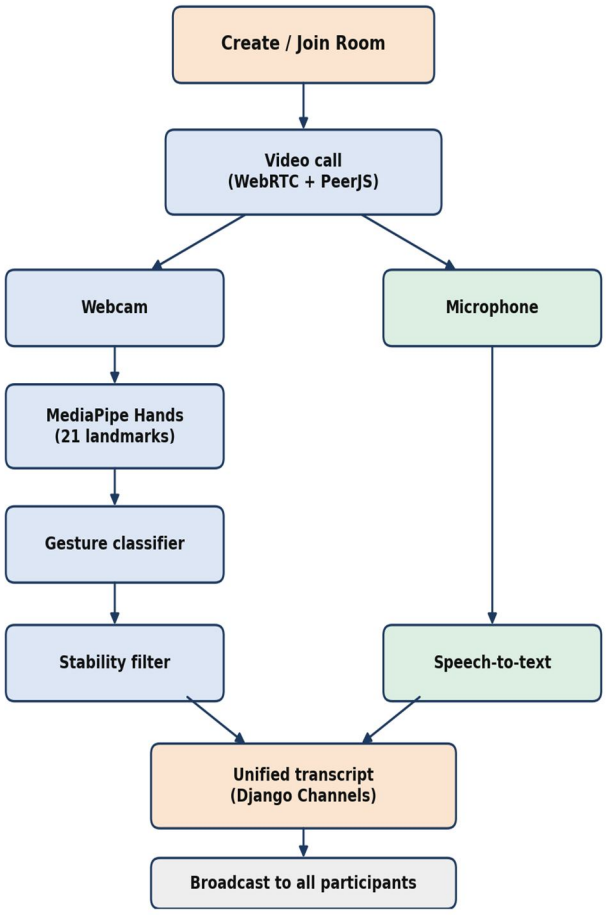


Fig. 1 Overall workflow of SignMeet AI

Fig. 1 shows the two parallel paths of the study. The sign path runs from the webcam through MediaPipe Hands, the gesture classifier and the stability filter; the speech path runs from the microphone through speech-to-text. Both merge into one unified transcript that Django Channels broadcasts to all participants, and the evaluation measures are taken on these same paths.

G. System Architecture Diagram

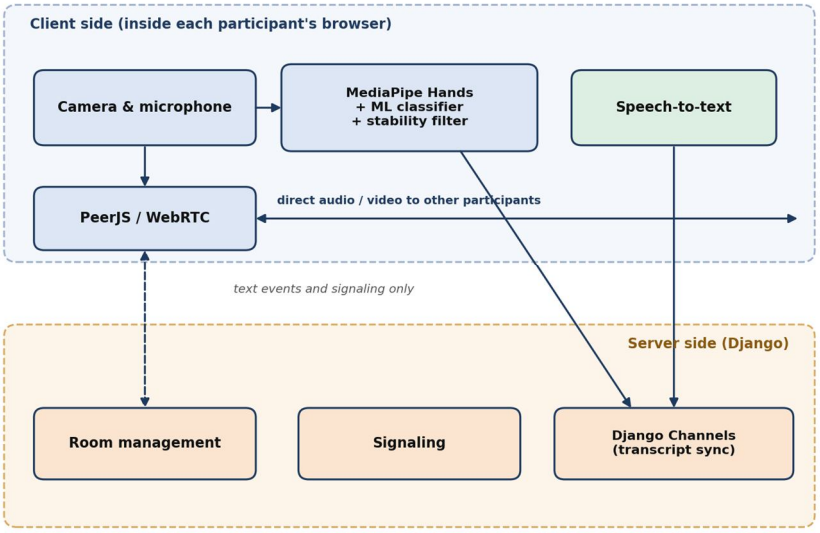


Fig. 2 Proposed architecture of SignMeet AI

Fig. 2 shows where each part runs. Recognition and media stay in the browser, and the Django server handles rooms, signaling and transcript synchronization. This split is what keeps the server light and the recognition private.

IV. PROPOSED WORK

A. Proposed Framework

SignMeet AI is a real-time framework that lets people who sign and people who speak share one video meeting. It combines peer-to-peer video calling, ASL gesture recognition, speech-to-text, a filter that steadies the recognized gestures, and a live transcript that every participant can read. The aim is plain: a signer and a speaker should take part in the same conversation without a human interpreter in the room. To get there, the system does most of its work on the user's own device and uses the server only to keep participants connected and in sync.

The system has five main components, listed in Table IV and shown together in Fig. 2.

TABLE IV
COMPONENTS OF SIGNMEET AI

Component	What it does	Main technology
Peer-to-peer video conferencing	Sends audio and video directly between participants	WebRTC, PeerJS
Real-time ASL gesture recognition	Finds the hand, extracts 21 landmarks and classifies the gesture	MediaPipe Hands, ML classifier
Gesture stability filtering	Stops single-frame errors from reaching the transcript	Confidence threshold, frame check, debouncing
Unified shared transcript	Merges sign and speech text and shares it with everyone	Django Channels, WebSocket
Room-based session management	Lets people create and join meetings with a room ID	Django, PeerJS

B. Algorithm

Algorithm 1: Per-Participant Processing in SignMeet AI

Input: a room ID, a webcam stream and a microphone stream. Output: a shared transcript.

1. Create or join a room using a room ID.
2. Request camera and microphone permission and capture the local stream.
3. Connect to the other participants through PeerJS and WebRTC.
4. For each webcam frame, detect the hand and extract the 21 landmarks.
5. Normalize the landmarks into a feature vector and classify it.
6. Pass the prediction through the stability filter (confidence check, N consecutive frames, debouncing).
7. In parallel, convert microphone speech into text.
8. Wrap each accepted gesture or speech result as a transcript event (speaker, type, text, timestamp) and send it to the server.
9. Broadcast the event through Django Channels and display it in every participant's transcript.

Two points clarify the algorithm. First, steps 4 to 6 run on the user's own device, so only the accepted result in step 8 leaves the browser. Second, steps 4 to 6 and step 7 run in parallel, so sign and speech never wait for each other.

C. Peer-to-Peer Video and Room Management

1) Peer-to-Peer Connection: WebRTC, built into modern browsers, lets two devices exchange audio and video directly [3]. Raw WebRTC leaves many connection details to the developer, so we use PeerJS on top of it for peer discovery and connection setup [23]. From the user's side, joining a call goes like this:

- A user creates a meeting and receives a unique room ID.
- The browser asks for permission to use the camera and microphone.
- The local audio and video stream is captured with the browser's media APIs.
- PeerJS uses the server briefly for signaling, which simply means helping the two browsers find each other and agree on how to connect.
- Once connected, audio and video flow directly from one participant to the other.
- Each browser displays the remote streams it receives.

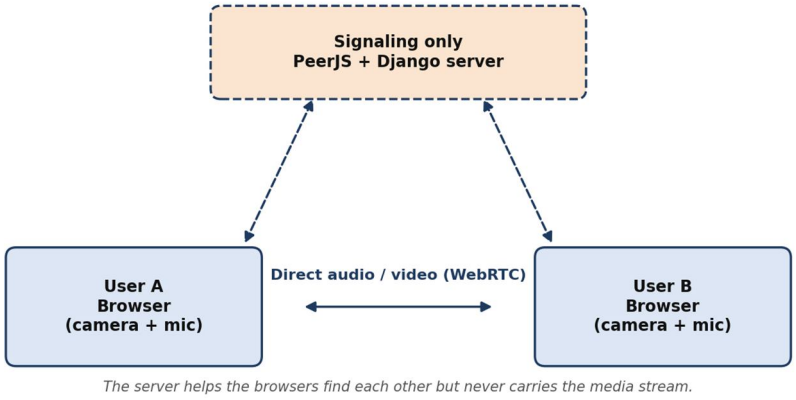


Fig. 3 Peer-to-Peer connection with the server used only for signaling

Because the Django server never carries the raw media, it needs far less bandwidth and processing power, which should keep delay low. Pure peer-to-peer calls do get heavier as a room grows, since each participant sends a separate stream to every other participant, so the prototype targets small rooms.

2) Room Creation and Joining: Meetings are organized around a room ID, not user accounts. When someone chooses Create Meeting, the backend generates a unique identifier, for example SM-7F4K92, which can be shared with anyone who should attend. To join, a participant types in the ID. The system checks that the room exists and that entry is allowed, and the WebRTC connection is then set up as in Section IV-C (Fig. 4).

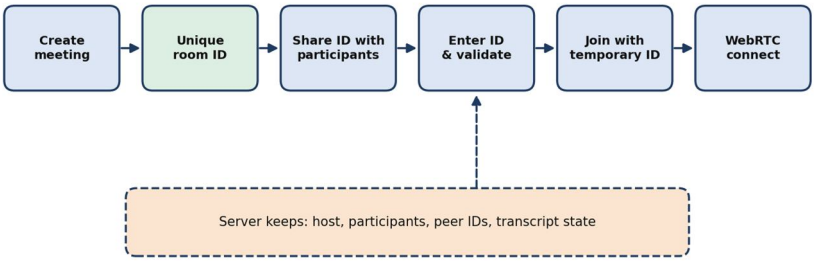


Fig. 4 Creating and joining a meeting by room ID

The server keeps only lightweight state for each room: the host, the participant list, their peer IDs and the transcript synchronization state. The prototype has no permanent accounts. Each person receives a temporary identity on entering, such as Participant 1 or Participant 2. That removes the sign-up step, which helps first-time users, but it is also why security is listed as a limitation in Section VIII.

D. Gesture Recognition Logic

The camera feed is analyzed on the user's own device, and frames are not sent to a server for recognition. Each frame passes through the stages shown in Fig. 5.

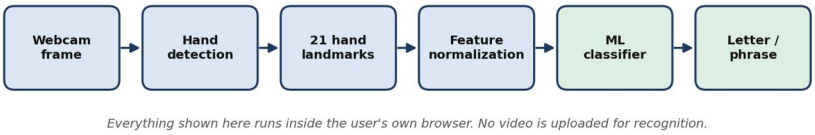


Fig. 5 ASL recognition pipeline running in the browser

1) *Camera input:* The browser reads the webcam stream and hands each frame to the MediaPipe Hands pipeline [2].
2) *Hand detection and landmarks:* MediaPipe Hands finds the hand in the frame and marks 21 landmarks: the wrist and the joints of the thumb and the four fingers. A busy image becomes a compact description of the hand's shape, so the classifier never has to look at the whole picture.
3) *Feature extraction:* The landmark coordinates are placed in one list of numbers, the feature vector:

$$F = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_{21}, y_{21}, z_{21}]$$

It holds 63 values (21 landmarks × 3 coordinates). The coordinates are normalized relative to the wrist or the hand's bounding box, so the same sign is recognized whether the hand is small or large, near the camera or far from it, and wherever it sits in the frame.

4) *Gesture classification:* The feature vector goes to a trained classifier, which maps the hand shape to an ASL alphabet letter or a supported phrase.

Local recognition has two clear benefits. Raw video is not uploaded for recognition, which helps privacy, and there is no network round trip, which helps speed. The classifier is a machine-learning model trained on these landmark features, and its vocabulary is limited to a defined set of static ASL letters and phrases (Section VIII).

E. Gesture Stability Logic

A classifier looks at every frame separately, so even a steady sign can produce a jumpy output. A typical stream might read A, A, B, A, A, C, A while the signer's hand never moves. Small hand movements, camera noise, lighting changes, slight shifts in finger position and brief detection errors all cause this. If every raw prediction went straight into the transcript, the text would be unreadable. Accuracy alone does not make real-time translation usable, so a filter sits between the classifier and the transcript. It applies three checks:

- Temporal stability. The system keeps a short history of recent predictions and accepts a gesture only if the same prediction appears in N consecutive frames. With N = 5, the stream A, A, A, A, A gives A, while A, B, A, C, A gives nothing.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

- Confidence threshold. The prediction must also be confident enough: P(g) > T, where P(g) is the model's confidence in gesture g and T is a threshold that we choose. A gesture is accepted only when both the stability and the confidence conditions are met.
- Debouncing. After a gesture is accepted, a short cool-down (or a requirement that the gesture change first) stops the same letter from being added again and again. Holding A for a second should give A, not AAAAA.

The threshold T and the cool-down length are tunable parameters of the filter: raising either gives steadier text at the cost of a slightly longer wait.

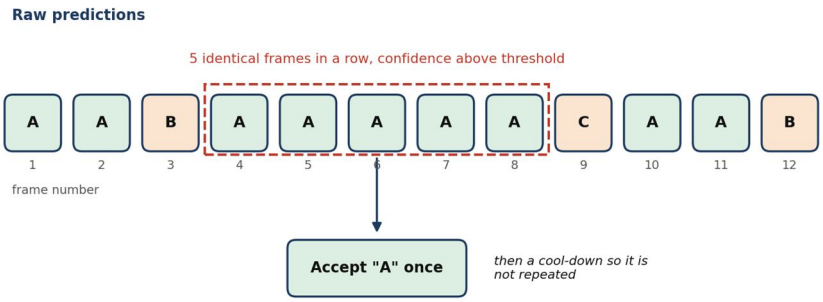


Fig. 6 Stability filter accepting a gesture only after five identical, confident frames

Together these checks cut false positives, stop repeated characters and smooth out unstable predictions, which keeps the transcript readable. The filter has a cost: a gesture appears only after N frames have agreed, so a larger N means steadier text but a longer wait.

Algorithm 2: Gesture Stability Filter

Input: a prediction, its confidence, the current time. Output: an accepted gesture or nothing.

1. Add the prediction to a short history and keep only the last N entries.
2. If the confidence is not above the threshold T, reject the prediction.
3. If fewer than N predictions are available, or they are not all the same gesture, reject.
4. If the gesture equals the last accepted one and the cool-down has not passed, reject.
5. Otherwise accept the gesture, record it and the time, and pass it to the transcript.

F. Shared Transcript Logic

Two independent pipelines feed the transcript (Fig. 7). The sign pipeline takes the webcam through MediaPipe Hands, the classifier and the stability filter. The speech pipeline takes the microphone through speech recognition. Whichever pipeline produces a result, it is packaged as a transcript event and sent to the server over a WebSocket connection, and Django Channels broadcasts it at once to everyone in the room.

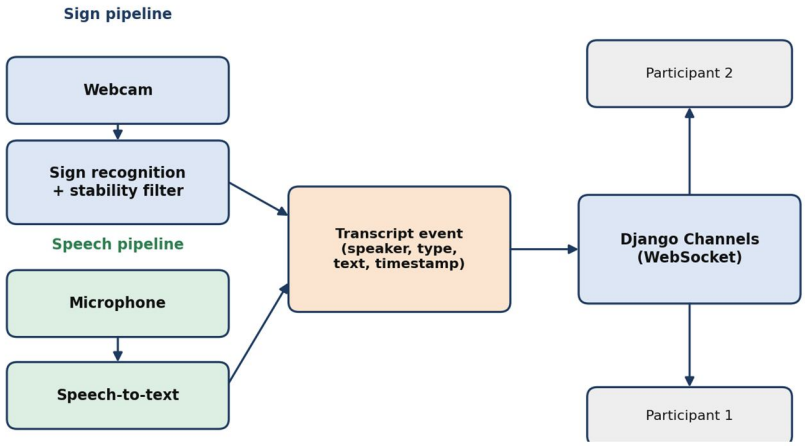


Fig. 7 Sign and speech pipelines merging into one transcript

A shared transcript might then read:

[Participant 1, Sign] Hello
[Participant 2, Speech] How are you?
[Participant 1, Sign] I am fine

Because signing and speaking participants see the same conversation, nobody needs a separate history or a separate window. The result is one transcript with real-time synchronization, speaker attribution and a consistent record for everyone.

G. Implementation Code

The two listings below are short sketches of the logic described above. They illustrate the stability filter and the transcript event and are not the full prototype source.

Listing 1 Python-style sketch of the gesture stability filter

```
history = []   # last N predictions
last, t = None, 0

def accept(pred, conf, now):
    global last, t
    history.append(pred)
    del history[:-N]
    if conf <= T:
        return None
    if len(history) < N:
        return None
    if any(g != pred for g in history):
        return None
    if pred == last and now - t < COOLDOWN:
        return None
    last, t = pred, now
    return pred
```

Listing 2 Example transcript event (illustrative values)

```
{
  "speaker": "Participant 1",
  "type": "sign",
  "text": "Hello",
  "timestamp": "2026-11-10T10:15:32Z"
}
```

In Listing 1, N, T and COOLDOWN are the tunable parameters of the filter. Listing 2 shows the four fields that every transcript event carries.

H. Experimental Variables

TABLE V
EXPERIMENTAL VARIABLES

Type	Variable	Description
Independent	Stability filter	Off (raw frame-by-frame output) or on.
Independent	Gesture category	Static alphabet gestures, common gestures, gestures with similar hand shapes.
Dependent	Recognition accuracy	Share of gestures identified correctly.

Type	Variable	Description
Dependent	Unwanted predictions and repeated characters	Counts per test session, with and without the filter.
Dependent	Sign-to-text and speech-to-text delay	Time before text appears in the transcript.
Controlled	Prototype and equipment	The same prototype, with a standard webcam and microphone.

I. Implementation Environment and Reproducibility

The prototype is a web application. Peer-to-peer audio and video use WebRTC with PeerJS; hand landmarks come from MediaPipe Hands running in the browser; a machine-learning classifier maps the landmarks to gestures; and the server side uses Django with Django Channels and WebSockets for rooms and the shared transcript. Speech-to-text runs alongside sign recognition on the participant's side.

To reproduce an evaluation, the following should be recorded with the results: the filter parameters N, T and the cool-down length, the gesture vocabulary used, the number of sessions and signers, the camera and lighting conditions, and the network conditions for the delay measurements. These records are what allow a later study to compare its figures with those reported here.

V. RESULTS AND DISCUSSION

A. Results Reporting Principle

This section reports how the prototype performed in testing: what worked, how accurate recognition was, what the stability filter changed and how long text took to appear. The accuracy, filter and delay figures come from sample tests of the prototype, so they should be read as indicative of its behavior and not as final measurements.

The prototype was tested as a working meeting application with a standard webcam and microphone. Recognition accuracy was compared across three categories: static alphabet gestures, common gestures, and gestures with similar hand configurations. The classifier is the landmark-based model described in Section IV-D. The figures reported below come from sample test sessions and are indicative: this study does not report a controlled benchmark with a fixed vocabulary, repeated trials and a larger group of signers, which is left to future work.

B. Functional Results

Table VI lists what each component did in the prototype. These are pass-or-fail functional checks, not measurements.

TABLE VI
FUNCTIONAL RESULTS BY COMPONENT

System component	Result
Peer-to-peer video communication	Calls were established between participants
MediaPipe hand detection	Hand landmarks were detected in webcam frames
ASL gesture classification	Supported gestures were recognized
Stability filtering	Unstable predictions were reduced (Section V-D)
Speech-to-text	Spoken words were converted to text
Unified transcript	Sign and speech output appeared in one view
Real-time synchronization	Transcript events reached all participants
Room management	Users could create and join rooms

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

C. Recognition Results

Table VII and Fig. 8 give the accuracy for each gesture category.

TABLE VII
RECOGNITION ACCURACY BY GESTURE CATEGORY

Gesture category	Accuracy
Static alphabet gestures	91%
Common gestures	88%
Similar hand configurations	82%
Overall	87%

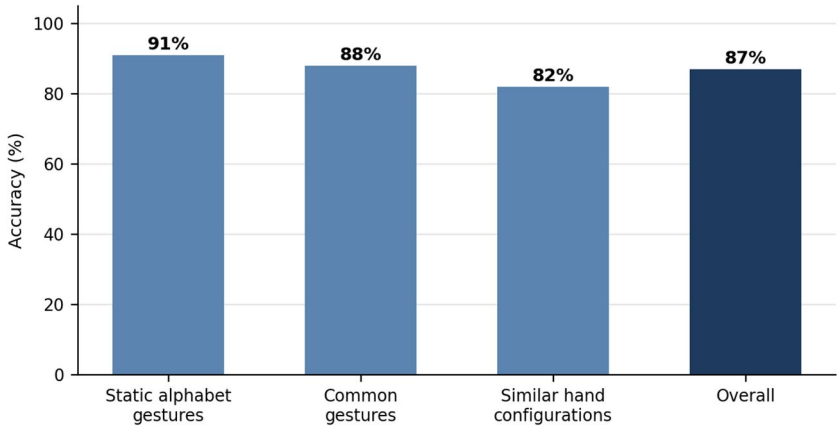


Fig. 8 Recognition accuracy by gesture category

Static, visually distinct gestures were the easiest to recognize at 91%. Common gestures reached 88%. Gestures with similar hand shapes were the hardest at 82%, because their landmark patterns overlap and the classifier can confuse one for another. The overall figure of 87% equals the unweighted mean of the three categories.

D. Stability Filtering Results

Table VIII and Fig. 9 compare the output with and without the filter.

TABLE VIII
EFFECT OF THE STABILITY FILTER

Parameter	Before filtering	After filtering
Unwanted predictions	18	6
Repeated characters	24	8
Stable recognition	78%	91%

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

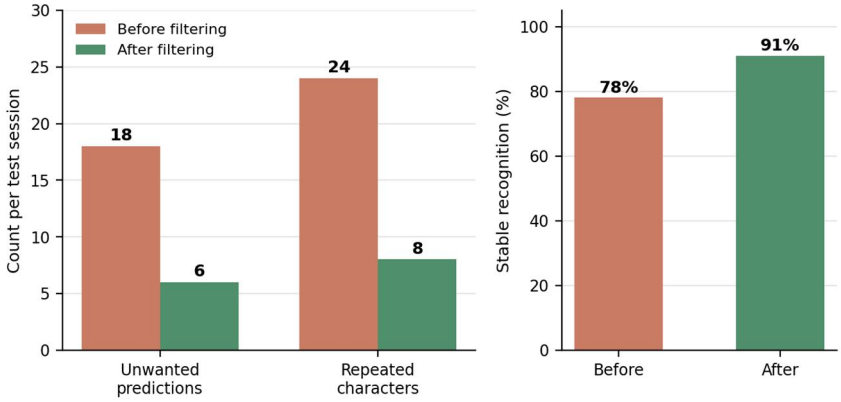


Fig. 9 Effect of the stability filter on false predictions and stable recognition

With the filter on, unwanted predictions fell from 18 to 6 and repeated characters from 24 to 8 per test session, a drop of two thirds in both cases. The share of stable recognition rose from 78% to 91%. Not every frame-level guess now becomes a transcript entry, which is what makes the output readable. The result points in the direction H1 predicts. Testing H1 properly would take repeated sessions and a significance test, which are not reported here.

E. Communication and Transcript Performance

Table IX and Fig. 10 report the delays and the state of synchronization.

TABLE IX
COMMUNICATION AND TRANSCRIPT PERFORMANCE

Parameter	Result
Average sign-to-text delay	About 320 ms
Average speech-to-text delay	About 342 ms
Transcript synchronization	Worked in the tests
Peer-to-peer communication	Worked in the tests

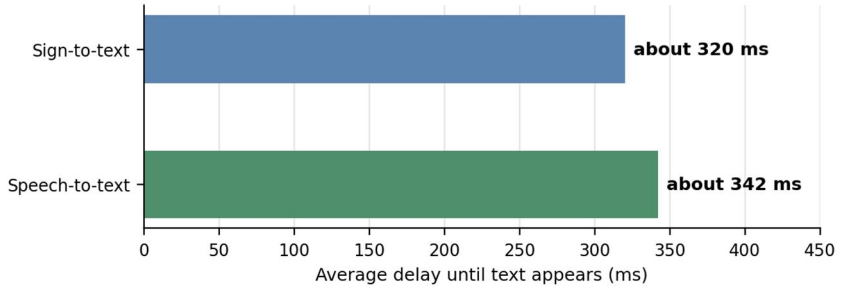


Fig. 10 Average delay before text appears

Text appeared after about 320 ms for signs and 342 ms for speech. Because WebRTC avoids continuous server-side media processing, the server stays light while Django Channels keeps the text in sync across participants. Part of the sign-to-text delay is the wait for N frames to agree: at 30 frames per second, five frames take about 170 ms. The wait for N frames is therefore a visible part of the sign-to-text delay. As a reference point, ITU-T Recommendation G.114 [24] treats one-way delays up to 150 ms as acceptable for most applications and delays up to 400 ms as acceptable provided the effect on quality is understood. The recommendation concerns voice, so it is used here only as a rough guide. Both measured delays fall in the 150 to 400 ms band, which is consistent with H2 for the sample tests but not conclusive.

F. Discussion

The results suggest that local gesture recognition, peer-to-peer communication and a real-time shared transcript are a sound base for an accessible video-conferencing platform. Three observations stand out:

- The stability filter matters. Raw frame-by-frame output is not something a person can read in a conversation. Filtering turned it into text that is usable, at the price of a short wait.
- Local processing keeps the server light. Recognition runs on the user's device and only small text events reach the server, which also means raw video is not uploaded for recognition.
- Similar hand shapes are the main accuracy problem. The 9-point gap between static alphabet gestures and similar configurations shows where more training data or extra features would help most.

The results also have limits. The figures come from a small sample and have no significance test. H3, that a shared transcript helps participants follow the conversation, has no participant data behind it yet. The prototype recognizes static gestures only, so the results say little about continuous signing at natural conversation speed.

Table X sets the hypotheses against what the sample tests show.

TABLE X
INTERPRETATION OF THE SAMPLE RESULTS AGAINST THE HYPOTHESES

Hypothesis	Evidence from the sample tests	Reading
H1	Unwanted predictions fell from 18 to 6 and repeated characters from 24 to 8 per session with the filter on.	Points in the direction H1 predicts; a significance test is not reported.
H2	Sign-to-text delay of about 320 ms and speech-to-text delay of about 342 ms.	Consistent with H2 within the 150 to 400 ms band used as a rough guide; not conclusive.
H3	No feedback from participants was collected.	Open; needs a study with participants.

On the sample tests, the prototype ran a peer-to-peer meeting, recognized static ASL gestures with 87% overall accuracy, reduced unstable output with the filter and delivered text in roughly a third of a second. Testing with a larger and more varied group of signers is needed before these findings can be generalized.

G. Hypothesis-Testing Plan

A fuller test of the hypotheses would extend the sample tests. H1 would be tested with repeated sessions and a significance test on the counts with and without the filter. H2 would be tested with delay measurements repeated over varied devices and networks. H3 would need feedback from signing and speaking participants who use the shared transcript, compared with their experience of separate or one-directional tools. Until then, the results in this section remain indicative.

VI. FINDINGS AND SUGGESTIONS

A. Findings

- Peer-to-peer communication suits this platform. WebRTC carries audio and video without the application server having to process the media.
- MediaPipe Hands is a practical way to detect hands in real time. Its landmark output cuts down the visual information the classifier must handle.
- Stability filtering is essential for usable translation. Classifying frame by frame produces repeated or wrong predictions, and the filter made the output readable in the sample tests.
- A single shared transcript puts signed and spoken words in one place. Whether it improves understanding in practice still needs testing with participants.
- Browser-based processing reduces reliance on remote services. It lowers network traffic for recognition and adds a privacy benefit for video.
- Room IDs reduce friction. Participants can join without creating an account.
- Recognition depends on conditions. Lighting, camera position, hand orientation, background and individual signing style all affect performance.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

B. Suggestions

- Dataset: Increase the size and diversity of the gesture dataset, including different users, hand sizes, signing styles and lighting conditions.
- Similar gestures: Improve recognition of signs with similar hand configurations.
- Continuous signing: Move from isolated gestures to continuous sign language recognition.
- Sentence building: Use contextual language models to turn recognized gestures into meaningful sentences.
- Language coverage: Add support for Indian Sign Language.
- Speech: Improve speech recognition in noisy environments.
- Security: Introduce optional authentication and secure meeting links for production use.
- Devices: Optimize the model for smartphones and low-resource devices.

C. Implications for Practice

For organizations that run online meetings, the prototype suggests that sign recognition and speech transcription can be offered inside one lightweight, account-free meeting tool, without separate software or a human interpreter for every call. For developers, it shows that recognition can run in the browser on landmarks, keeping the server light and raw video on the user's device. For users, a single shared transcript means signing and speaking participants follow one conversation.

In summary, this paper presented SignMeet AI, a browser-based video-conferencing prototype that combines peer-to-peer WebRTC video, in-browser ASL recognition from 21 MediaPipe hand landmarks, a temporal stability filter, speech-to-text and a Django Channels transcript that every participant sees. In the sample tests, the filter reduced unwanted predictions and repeated characters by about two thirds, overall recognition accuracy was 87%, and text appeared after roughly a third of a second. These figures are indicative only, and continuous signing, Indian Sign Language support and a user study with signers remain future work.

VII. FUTURE SCOPE

SignMeet AI is a starting point, and it can grow in several directions.

1) *Continuous sign language translation.* The prototype recognizes single gestures or letters. Temporal models such as long short-term memory (LSTM), gated recurrent unit (GRU) or Transformer architectures could extend it to full sentences by learning sequences of gestures.
2) *Bidirectional communication.* Later versions could add speech-to-sign animation and spoken output of recognized signs, so that both people communicate in the form they prefer.
3) *Indian Sign Language support.* Extending the system to ISL would make it far more useful to Indian users and educational institutions.
4) *Multilingual translation.* Recognized sign content could be translated into several spoken languages, for example English, Hindi and Marathi.
5) *AI-based contextual translation.* A language model could use the order and context of gestures to produce grammatical sentences, instead of translating each gesture alone.
6) *Sign-language avatar.* An animated 3D avatar could turn spoken or typed sentences into sign language.
7) *Mobile application.* With lighter computer-vision models, the system could be optimized for Android and iOS devices.
8) *Emotion and facial expression recognition.* Sign language also relies on facial expression and body movement. Adding face and pose recognition would capture meaning more fully.
9) *Enhanced security.* A production version needs stronger protection:

- User authentication
- Password-protected rooms
- Expiring meeting links
- Host controls and participant permissions
- Secure WebSocket connections
- Stronger privacy controls

VIII. LIMITATIONS OF THE STUDY

SignMeet AI shows promise, but its limits should be stated plainly. Table XI summarizes them, and the paragraphs after it explain the ones that need more than a line.

TABLE XI
LIMITATIONS, THEIR EFFECT, AND A POSSIBLE WAY FORWARD

Limitation	Effect	Possible way forward
Limited sign vocabulary	Cannot cover natural ASL	Expand the dataset and vocabulary
Static gestures only	Struggles with signs that involve movement	Use temporal models
Environmental conditions	Accuracy drops in poor lighting or clutter	Train on varied conditions
Similar gestures	Visually close signs get confused	Add more distinguishing features
Device performance	Lower frame rate on weak devices	Optimize or compress the model
Speech recognition quality	Errors in noisy or accented speech	Noise handling, better models
Network dependency	Call quality varies with connection	Adaptive quality, fallback relays
Prototype security	Not ready for production use	Authentication and secure links
Limited evaluation	Results cannot yet be generalized	Test with more diverse users

1) *Scope of the prototype.* The prototype supports a limited set of ASL letters and phrases, so it cannot represent the vocabulary or grammar of natural ASL. Recognizing a fixed hand shape is much easier than recognizing continuous signing, where movement, orientation, timing and transitions all matter. Some signs also have almost the same hand configuration, so a landmark-based classifier can mistake one for another.
2) *Conditions and devices.* Recognition can be affected by poor lighting, camera quality, background objects, hand occlusion, distance from the camera and hand orientation. Because recognition runs in the browser, slower devices may see a lower frame rate or slower inference. WebRTC reduces server-side media handling, but call quality still depends on bandwidth, network stability, network address translation (NAT) traversal and device connectivity.
3) *Speech recognition.* Speech-to-text accuracy can fall because of background noise, pronunciation, accents, microphone quality and people speaking at the same time.
4) *Security and evaluation.* Joining by room ID with no login suits a prototype, but real-world use would need stronger security. The evaluation is also small: the system needs to be tested with a larger and more varied group of sign-language users before its effectiveness can be generalized.

IX. APPENDIX

A. Summary of Sample Results

The sample results reported in Section V are collected in Table XII for reference.

TABLE XII
SUMMARY OF SAMPLE RESULTS

Result	Value	Reference
Overall recognition accuracy	87%	Table VII, Fig. 8
Unwanted predictions (before / after filtering)	18 / 6	Table VIII, Fig. 9
Repeated characters (before / after filtering)	24 / 8	Table VIII, Fig. 9
Stable recognition (before / after filtering)	78% / 91%	Table VIII, Fig. 9
Average sign-to-text delay	About 320 ms	Table IX, Fig. 10
Average speech-to-text delay	About 342 ms	Table IX, Fig. 10
Transcript synchronization	Worked in the tests	Table IX

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

B. Research Integrity and Acknowledgment Note

The recognition, filter and delay figures in this paper come from sample tests of the prototype and are presented as indicative, not as a controlled benchmark. No claim is made about participant experience, because no participant feedback was collected. The authors thank their guide, Prof. Deepika Sarwate, and the Department of Master of Computer Applications, JSPM's Narhe Technical Campus, Savitribai Phule Pune University, for their guidance and support during this work.

C. Glossary of Terms and Abbreviations

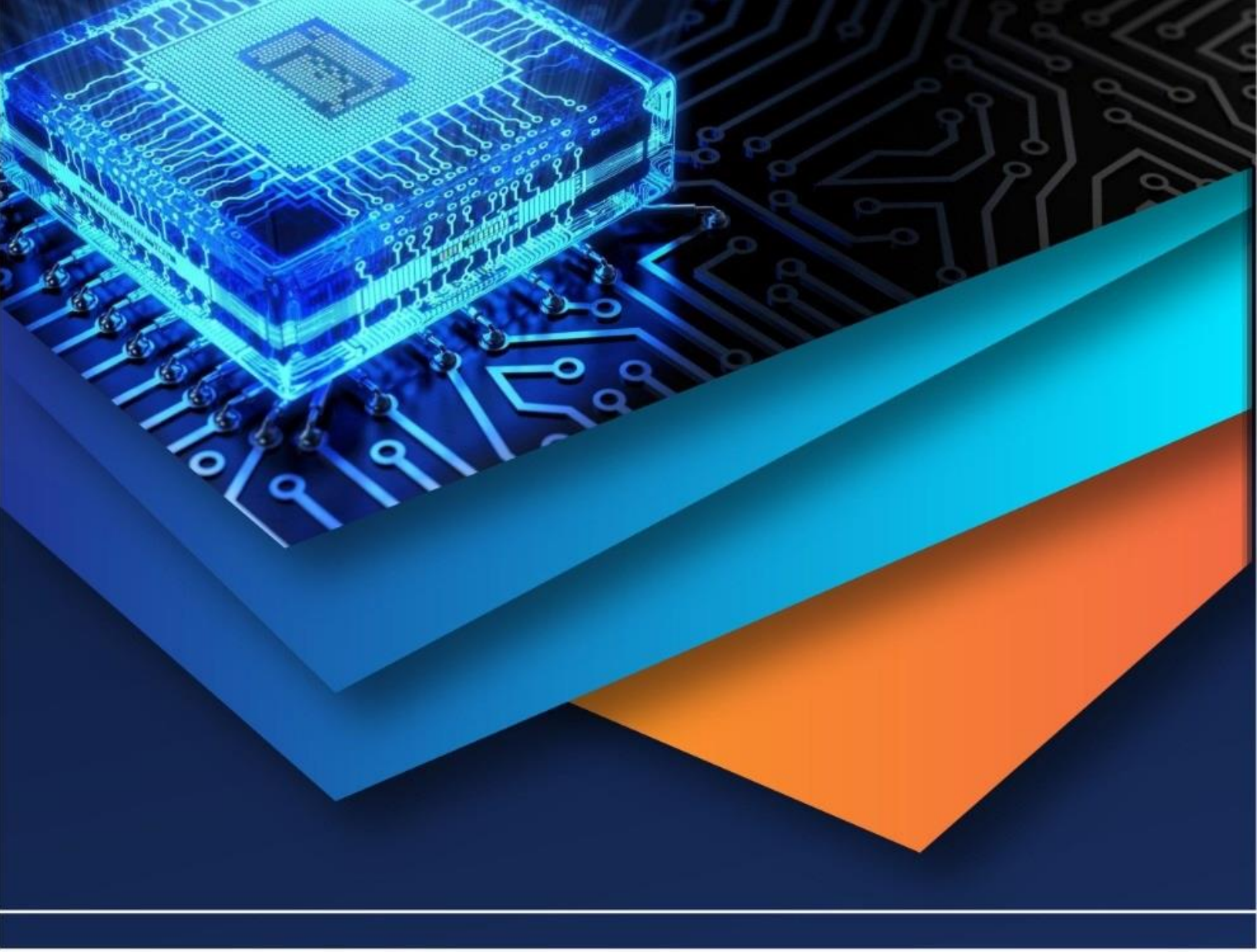
TABLE XIII
GLOSSARY OF TERMS AND ABBREVIATIONS

Term	Meaning
ASL	American Sign Language
Bi-LSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
ISL	Indian Sign Language
LSTM	Long Short-Term Memory
ML	Machine Learning
NAT	Network Address Translation
WebRTC	Web Real-Time Communication
WHO	World Health Organization
XR	Extended Reality

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

REFERENCES

[1] World Health Organization, World Report on Hearing. Geneva, Switzerland: WHO, 2021. [Online]. Available: https://iris.who.int/items/333af8d0-a4f3-4a5e-880d-9bd530f6f469
[2] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "MediaPipe Hands: On-device real-time hand tracking," arXiv:2006.10214, Jun. 2020.
[3] W3C, "WebRTC 1.0: Real-time communication between browsers," W3C Recommendation, Jan. 2021. [Online]. Available: https://www.w3.org/TR/webrtc/
[4] A. Sangeetha Priya, S. Aneesha Barveen, and M. Shahar Banu, "A real time webcam based sign language translation system using computer vision," Shanlax Int. J. Arts, Sci. Humanit., vol. 13, special issue 3, pp. 225–230, Feb. 2026, doi: 10.34293/sijash.v13iS3-i1-Feb.10285.
[5] S. O. Aabid, "Real-time sign language translator for specially abled," Int. J. Res. Appl. Sci. Eng. Technol., 2025, doi: 10.22214/ijraset.2025.71570.
[6] M. N. Lokhande, A. Pujar, K. R. Pardeshi, T. A. Shah, Y. N. Shah, and Y. S. Pandhare, "Real-time sign language translator," in Proc. 2025 Int. Conf. Sustain. Technol. Humanity Smart World (HSWTech), 2025, pp. 1–6, doi: 10.1109/HSWTech64936.2025.11278130.
[7] B. Sonare, A. Padgal, Y. Gaikwad, and A. A. Patil, "Video-based sign language translation system using machine learning," in Proc. 2021 2nd Int. Conf. Emerging Technol. (INCET), 2021, pp. 1–4, doi: 10.1109/INCET51464.2021.9456176.
[8] A. Moryossef, I. Tsochantaridis, R. Aharoni, S. Ebling, and S. Narayanan, "Real-time sign language detection using human pose estimation," Lecture Notes Comput. Sci., 2020, doi: 10.1007/978-3-030-66096-3_17.
[9] B. S. Natarajan, R. E., R. Elakkiya, K. Kotecha, A. Abraham, L. Gabralla, and V. Subramaniyaswamy, "Development of an end-to-end deep learning framework for sign language recognition, translation, and video generation," IEEE Access, vol. 10, pp. 104358–104374, 2022, doi: 10.1109/ACCESS.2022.3210543.
[10] M. Vishwekar, D. Injeti, R. O. Gupta, and N. Gowalker, "SMART SIGN: Live sign language interpretation for barrier-free video conferencing," Int. J. Multidiscip. Res., vol. 7, no. 1, 2025, doi: 10.36948/ijfmr.2025.v07i01.36374.
[11] S. Mukherjee, M. Akhtar, and R. Kannadasan, "Real time captioning of sign language gestures in video meetings," arXiv:2507.14543, 2025, doi: 10.48550/arXiv.2507.14543.
[12] B. Dey and R. Paul, "Integrating Indian Sign Language recognition with real-time speech synthesis for video conferences," Int. J. Sci. Res. Eng. Manag., 2025, doi: 10.55041/ijsrem42865.
[13] J. Pradeep, M. Harikrishnan, J. Gopinathan, and B. Jayasuriya, "AI-powered bidirectional sign language translator for real-time mobile voice calls," in Proc. 2026 Int. Conf. Comput. Netw. Inventive Commun. Technol. (ICCNCT), 2026, pp. 792–798, doi: 10.1109/ICCNCT68477.2026.11590429.
[14] B. K. Nayak, B. K. Giri, and S. Samantaray, "Real-time sign language chat application with speech output," Amer. J. AI Digit. Transformation Regenerative Pharmacist, 2026, doi: 10.64751/dszspw04.
[15] S. C., "Inclusive communication: Leveraging AI for sign language translation and real-time audio transcription," Int. J. Sci. Res. Eng. Manag., 2025, doi: 10.55041/ijsrem48555.
[16] Y. F. Said, S. Boubaker, S. M. Altowaijri, A. Alsheikhy, and M. Atri, "Adaptive Transformer-based deep learning framework for continuous sign language recognition and translation," Mathematics, vol. 13, no. 6, p. 909, 2025, doi: 10.3390/math13060909.
[17] J. R., K. M., and S. T., "Real-time multilingual dynamic sign language translation system using Vision Transformer," in Proc. 2026 4th Int. Conf. Inventive Comput. Informat. (ICICI), 2026, pp. 925–930, doi: 10.1109/ICICI68773.2026.11581075.
[18] A. Shinde and R. Kannan, "Sign language video translation for effective communication using hybrid deep learning techniques," J. Intell. Decis. Making Inf. Sci., vol. 3, 2026, doi: 10.59543/jidmis.v3.475.
[19] N. Tantaroudas, A. J. McCracken, I. Karachalios, and E. Papatheou, "INTERACT: An AI-driven extended reality framework for accessible communication featuring real-time sign language interpretation and emotion recognition," arXiv:2604.05605, 2026, doi: 10.48550/arXiv.2604.05605.
[20] N. Tantaroudas, A. J. McCracken, I. Karachalios, and E. Papatheou, "INTERACT: AI-powered extended reality platform for inclusive communication with real-time sign language translation and sentiment analysis," Open Res. Europe, 2026, doi: 10.12688/openreseurope.23201.1.
[21] S. Srivastava, S. Singh, S. Goswami, M. Gupta, and P. Garg, "SignSpeak: A real-time deep learning framework for bidirectional sign language translation," in Proc. 2026 6th Int. Conf. Emerging VLSI Semicond. Technol. AI Comput. Appl. (EVST), 2026, pp. 1–7, doi: 10.1109/EVST69093.2026.11660646.
[22] H. Waheed, G. M. E. Abro, S. Memon, and N. Saeed, "SIGN-AIR: Real-time sign language recognition and translation using gesture-aware neural networks on aerial robotic platforms," Sci. Rep., 2026, doi: 10.1038/s41598-026-68713-1.
[23] PeerJS, "PeerJS: Simple peer-to-peer with WebRTC." [Online]. Available: https://peerjs.com/
[24] ITU-T, "One-way transmission time," Recommendation G.114, May 2003.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)