Software Requirements Specifications

Project Title: **SignLink**

Project Description:

A Real-Time Sign Language Translation and Subtitling System for Video Conferencing Platforms

Project Code:

SLTS-S07-R38

Internal Advisor:

Dr. Fahad Maqbool

Project Manager:

Dr. Muhammad Ilyas

Project Team:

Muhammad Hassan Shahbaz

Behlol Abbas

Submission Date:

19-10-2025

_____ _____

Supervisor's Signatures **Project Manager's Signatures**

Document Information

Category	Information
Customer	UOS
Project	SignLink
Document	Requirement Specifications
Document Version	1.0
Identifier	SLTS-S07-R38
Status	Draft
Author(s)	M.Hassan Shabaz , Behlol Abbas
Approver(s)	Dr. Muhammad Ilyas
Issue Date	Oct. 20, 2025
Document Location	https://github.com/devHassan007/FYP-SLTS
Distribution	1. **Advisor**: Dr. Fahad Maqbool 2. **PM:** Dr. Muhammad Ilyas 3. **Project Office**: Department Of CS & AI

Definition of Terms, Acronyms and Abbreviations

Term	Description
LSTM	Long-Short Term Memory
WLASL	World Level American Sign Language
TTS	Text To Speech
SDK	Software Development Kit
ASL	American Sign Language
BSL	British Sign Language
API	Application Programming Interface
SLR	Sign Language Recognition
VSL	View Sign Language
NLP	Natural Language Processing
GPU	Graphics Processing Unit

Table of Contents

1. Introduction

1.1 Purpose of Document

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements for the Sign Link software application. It serves as a comprehensive reference to guide the development, testing, and deployment processes, ensuring alignment with stakeholder expectations and project objectives. The document outlines the system's capabilities, constraints, and interfaces to facilitate clear communication among team members and external parties.

The intended audience includes:

1. **Project Team Members:** Developers, machine learning engineers, and testers responsible for implementing and validating the system.
2. **Project Managers and Stakeholders**: For oversight, resource allocation, and progress tracking.
3. **End Users and Representatives:** Members of the Deaf and hard-of-hearing community, accessibility experts, and potential integrators (e.g., video conferencing platform providers) to review usability and inclusivity aspects.
4. **Academic and Industrial Reviewers**: Researchers and corporate entities interested in AI-driven accessibility tools for evaluation and potential collaboration.

1.2 Project Overview

Sign Link is a software application designed to enhance accessibility in video conferencing for Deaf and hard-of-hearing individuals. It integrates with platforms like Zoom and Google Meet to provide real-time sign language recognition, translation to spoken English, and live subtitling. By leveraging human pose estimation (e.g., MediaPipe) and temporal deep learning models (e.g., LSTM or Transformer) trained on datasets such as WLASL and Multi-View Sign Language (Multi-VSL), the system extracts skeletal keypoints from video feeds, recognizes signs, and converts them into text and speech outputs.

The software will be used as a plug-and-play add-on in professional, educational, and social virtual meetings, allowing Deaf users to communicate directly without relying solely on human interpreters. Key goals include improving digital inclusivity, reducing communication barriers, and advancing real-time continuous sign language recognition. Benefits encompass academic contributions to AI accessibility, industrial applications for diverse team collaboration, and social empowerment by fostering equitable participation in remote interactions.

1.3 Scope

The project focuses on developing a prototype that enables real-time sign language translation for single users in controlled environments, with extensibility for future enhancements.

In-Scope:

1. Real-time detection and recognition of a subset of sign language vocabulary (100–200 common words/signs in American Sign Language (ASL), extensible to others like British Sign Language (BSL)).
2. Translation of recognized signs into English text, followed by text-to-speech synthesis and subtitle display.
3. Integration as a desktop application or browser extension with at least one major video conferencing platform (e.g., Zoom) via APIs or SDKs.
4. Support for single-signer scenarios in well-lit environments with clear backgrounds.
5. Data preprocessing, model training, and evaluation targeting >90% accuracy and <100ms latency per gesture.
6. Output in English audio (injected into the audio stream) and on-screen subtitles for all participants.

Out-of-Scope:

1. Full recognition of complex sign language grammar, non-manual markers (e.g., facial expressions), or regional dialects beyond core vocabularies.
2. Simultaneous translation for multiple signers.
3. Reverse translation from spoken language to sign language.
4. Development of a standalone video conferencing platform; the focus is on an integration module.
5. Robust handling of low-light, cluttered, or noisy environments in the initial prototype.
6. Hardware-specific optimizations (e.g., for mobile or wearable devices).
7. Multilingual support beyond English outputs in the prototype phase.

2. Overall System Description

Sign Link will be developed in an academic environment using open-source tools and university computing resources, then deployed as a user-friendly application for everyday video conferencing. It anticipates users from diverse backgrounds, with constraints emphasizing accessibility and efficiency.

2.1 User characteristics

The system caters to multiple user classes, prioritized by criticality:

1. Primary Users :

The non-technical users are the most critical users.

- **Deaf and Hard-of-Hearing Individuals:** End users who rely on sign language for communication. They require intuitive interfaces for setup and real-time feedback, with minimal technical expertise assumed (e.g., basic computer literacy).

- **Hearing Participants in Meetings**: Non-signers who benefit from translated speech and subtitles. They expect seamless integration without disrupting workflow.

2. Secondary Users :

The technical users are important but less critical than primary users.

- **Human Interpreters:** Professionals who may use the tool as a supplementary aid. They possess domain knowledge in sign language and require advanced configuration options.
- **Accessibility Coordinators/Educators:** In educational or corporate settings, these users manage deployments and need administrative features for customization.

3. Tertiary Users :

The supportive users like developers and researchers mean a lot.

- **Developers and Researchers**: For maintenance and extensions, requiring access to source code and documentation.
- **General Public:** Casual users in social contexts, with varying technical skills.
- **User classes distinguished by needs**: primary users focus on usability and reliability, while secondary/tertiary emphasize extensibility.

2.2 Operating environment

The software will operate on standard desktop or laptop computers, compatible with video conferencing applications. Key components include:

- **Hardware Platform:** Modern PCs with webcams, microphones, and speakers; GPU acceleration (e.g., NVIDIA CUDA-compatible) recommended for model inference.
- **Operating System:** Cross-platform support for Windows, macOS, and Linux.
- **Software Components:** Python 3.x runtime; libraries such as PyTorch/TensorFlow for ML models, OpenCV/MediaPipe for video processing, and WebRTC for browser integration. It must coexist with video conferencing tools like Zoom or Google Meet, using virtual cameras/audio cables (e.g., VB-Audio) for input/output routing.

2.3 System constraints

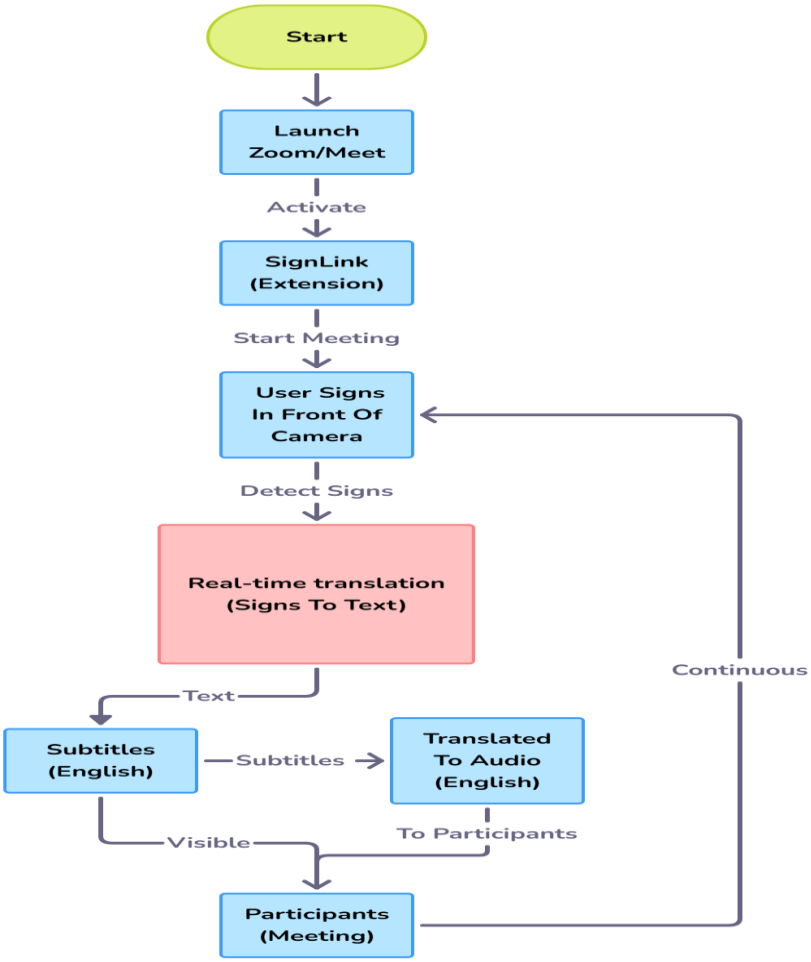
The system is subject to the following constraints:

- **Software Constraints:** Reliance on open-source libraries (e.g., MediaPipe, gTTS) limits features to available APIs; models must be optimized for real-time execution without proprietary enhancements.
- **Hardware Constraints:** Requires sufficient processing power (e.g., multi-core CPU/GPU) for low-latency inference; prototype assumes standard webcams (no specialized sensors).
- **Cultural Constraints:** Initial focus on ASL with English outputs; extensibility to other sign languages (e.g., BSL) but no support for non-Latin scripts in subtitles.

- **Legal Constraints:** Compliance with data privacy laws (e.g., GDPR for user video processing); privacy-preserving techniques (e.g., keypoint extraction over raw video storage) to avoid handling sensitive biometric data.
- Environmental Constraints: Designed for quiet, well-lit indoor settings; no audio alerts to accommodate noisy environments or hearing sensitivities.
- **User Constraints:** Interface prioritizes visual elements (e.g., graphical controls) over text for accessibility; assumes adult users but adaptable for educational contexts with simpler visuals.
- **Off-the-Shelf Components:** Datasets like WLASL impose vocabulary limits; integration SDKs (e.g., Zoom API) may restrict customization, transferring any API rate limits or version dependencies to the system.

3. External Interface Requirements

The system interfaces with external components via video/audio streams and APIs. A high-level context diagram would depict: User Video Feed → Sign Link (Processing) → Video Conferencing Platform (Outputs: Subtitles/Audio), with datasets and ML libraries as supporting inputs.



3.1 Hardware Interfaces

Hardware Interfaces include the hardware components like:

1. **Webcam/Video Input:** Supports standard USB/web-integrated cameras for capturing signer video at 720p+ resolution; data interaction involves frame extraction for pose estimation.
2. **Microphone/Speaker**: Virtual audio routing for injecting synthesized speech; no direct hardware control, but compatibility with built-in or external devices.
3. **Supported Devices:** Laptops/desktops; future extensibility to mobiles, but prototype focuses on stationary setups.

3.2 Software Interfaces

Software Interfaces include components like software applications , toolset and data.

1. **Video Conferencing Platforms**: Integration with Zoom/Google Meet SDKs (name: Zoom SDK v5.x+; version: latest stable). Purpose: Hook into video streams for input and overlay subtitles/output audio. Data exchanged: Video frames (input), text subtitles (output), audio packets (synthesized speech).
2. **ML Libraries and Datasets:** MediaPipe (for pose estimation), PyTorch/TensorFlow (for models); WLASL/Multi-VSL datasets (static files for training). Services: Feature extraction and inference; communications via API calls; shared data includes keypoints and model weights.
3. **TTS Engines:** Google gTTS or Pyttsx3; input: Translated text; output: Audio streams for injection.
4. **NLP Tools:** BERT-like models for sentence formation; integrated via libraries for contextual processing.

3.3 Communications Interfaces

Communication Interfaces refer to the mediums and components that are the main concern for a communication/connection over a network.

1. **Network Protocols:** WebRTC for browser-based real-time video/audio streaming; HTTP/HTTPS for API calls to external services (e.g., TTS).
2. **Message Formatting**: JSON for API payloads (e.g., subtitle overlays); synchronized timestamps for audio/subtitle alignment.
3. **Security/Encryption:** HTTPS for data transfers; no storage of raw video to ensure privacy; optional encryption for sensitive sessions.
4. **Data Transfer Rates:** Target <100ms end-to-end latency; synchronization via frame buffering to handle network variability.

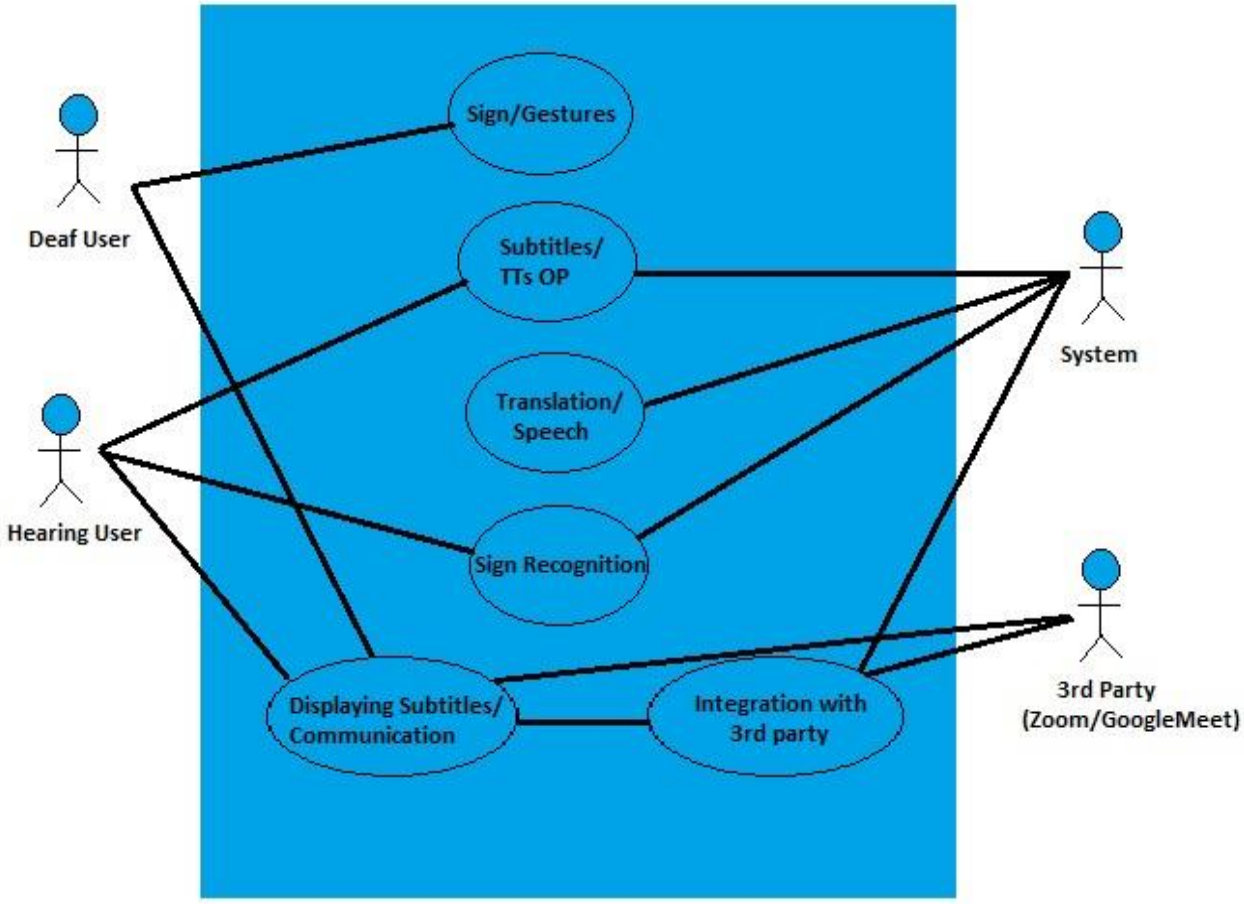
4. Functional Requirements

The system supports the following business events related to accessible video conferencing:

1. **Sign Detection and Recognition:** Capture video feed, extract keypoints, and recognize signs in real-time using trained models.
2. **Translation to Text:** Convert recognized glosses into coherent English sentences via NLP.
3. **Speech Synthesis:** Generate audio from translated text and inject into the conference audio stream.
4. **Subtitle Generation:** Display real-time text overlays for all participants.
5. **System Integration:** Hook into platform APIs for seamless input/output routing.
6. **Configuration and Setup**: Allow users to select sign language, calibrate video, and test outputs.
7. **Error Handling:** Detect low-confidence recognitions and provide fallback notifications (e.g., visual alerts).

4.1 Use Case Diagram

The use case diagram below explains two use cases discussed later.



Use Case-1

Gesture Recognition and Translation

Actors: Deaf User, System

Description:
The Deaf User performs sign language gestures in front of a camera. The system captures the video feed, preprocesses the data, and uses trained machine learning models to recognize the gestures. These gestures are then translated into corresponding text or speech output using a speech synthesis engine. This allows the Hearing User to understand the Deaf User's message in real time.

Steps:

1. Deaf User starts communication through the application.
2. System captures sign language input through the camera.
3. Preprocessing and recognition occur using the trained model.
4. Recognized gestures are converted to text or speech.
5. Hearing User receives the translated message.

Goal: Enable real-time understanding of sign language without a human interpreter.

Use Case-2

API Integration with Video Conferencing Platforms

Actors: System, Video Platform (Zoom/Google Meet)

Description:
The system integrates with third-party video conferencing platforms through an API to enable real-time translation within online meetings. This allows users to communicate seamlessly across different environments without needing a separate application.

Steps:

1. The system establishes a connection with the video platform using secure API keys.
2. When a meeting starts, the system begins monitoring both video and audio streams.
3. The system processes the Deaf User's gestures and the Hearing User's speech.
4. Translated text, speech are displayed within the video conferencing interface.
5. The session data is optionally logged for improving model accuracy in future updates.

Goal:
To provide smooth, cross-platform integration that ensures accessibility and inclusivity during online meetings without requiring additional setup from the users.

5. Non-functional Requirements

The system doesn't support the following business events related to accessible video conferencing:

5.1 Performance Requirements

1. **Speed/Latency:** <100ms per gesture recognition and translation.
2. **Accuracy/Precision:** >90% on benchmark datasets for word-level recognition.
3. **Capacity**: Handle continuous video streams up to 30 FPS; support meetings with up to 10 participants (subtitles/audio for all).
4. **Reliability:** 99% uptime in stable environments; graceful degradation in suboptimal conditions.
5. **Safety:** Minimal risk, as outputs are assistive; no critical dependencies.

5.2 Safety Requirements

The system poses low risk of harm, focusing on non-invasive video processing. Safeguards include:

- Prevention of audio feedback loops via virtual routing.
- No storage of personal video data.
- Compliance with accessibility standards (e.g., WCAG) to avoid misleading translations.

No specific safety certifications required, but adherence to ethical AI guidelines for inclusivity.

5.3 Security Requirements

1. **Privacy:** Process data locally where possible; use anonymized keypoints to avoid biometric storage.
2. **Integrity:** Secure API integrations with authentication tokens.
3. **Authorization:** User-level access controls for configuration; no shared data without consent.
4. **Compliance:** Align with privacy regulations (e.g., CCPA/GDPR); no certifications mandated for prototype.

5.4 User Documentation

- **User Manual:** PDF guide covering installation, setup, and troubleshooting.
- **Online Help:** In-app tooltips and FAQs.
- **Tutorials:** Video demos for configuration and usage, accessible via the application interface.

6. Assumptions and Dependencies

Sign Link assumes stable internet and compatible hardware (e.g., webcams) while depending on external datasets, third-party libraries and video conferencing SDKs for integration and functionality.

Assumptions:

1. Availability of stable internet for platform integrations (though local processing is prioritized).
2. Users have compatible hardware (e.g., webcams); signers are in well-lit environments.
3. Datasets like WLASL remain accessible and unchanged; ML models perform as benchmarked.
4. Future changes in sign language standards or platform APIs may require updates.

Dependencies:

1. External datasets (WLASL, Multi-VSL) for training.
2. Third-party libraries (MediaPipe, PyTorch) and their updates.
3. Video conferencing SDKs (e.g., Zoom) for integration; delays in their development could impact timelines.
4. University resources for GPU training; open-source tools for deployment.

7. References

Ref. No.	Document Title	Date of Release/ Publication	Document Source
SLTS-S07-R38	Project Proposal	Oct 13, 2025	https://github.com/devHassan007/FYP-SLTS