Software Engineering Department
Braude College

Capstone Project Phase A – 61998

Real Time Call Translation

25-2-D-5

Advisors:

Dr. Dan Lemberg, lembergdan@braude.ac.il

Mrs. Elena Kramer, elenak@braude.ac.il

Students:

Amir Mishayev, Amir.mishayev@e.braude.ac.il

Daniel Fraimovich, Daniel.fraimovich@e.braude.ac.il

Table of Contents

Abstract

This project addresses the high demand for effective real-time multilingual communication in multi-party calls, particularly given the limitations of current solutions in voice preservation and support for languages like Hebrew. To enable natural conversations with retained vocal identity, our research evaluated key AI technologies. We explored end-to-end speech-to-speech models, various open-source voice cloning libraries, Automatic Speech Recognition (ASR) systems, and Neural Machine Translation (NMT) back-ends. This thorough assessment showed that while advanced models exist, practical real-time use *requires* a modular architecture due to high resource demands and latency issues with integrated solutions like Meta's SeamlessM4T, and difficulties in Hebrew voice cloning due to limited training data. Because of this, our system design integrates Whisper for local ASR, Google Translate API for NMT, and Coqui xTTS for voice cloning experimentation. This research has created a strong plan for building a high-quality, audio-to-audio translation experience for users of Hebrew, English, and Russian.

1. Introduction

In an era defined by unprecedented global connectivity and an ever-expanding human population, now approximately 8 billion, the necessity for effective cross-lingual communication has never been more pronounced. The internet and digital technologies have woven societies into a complex global tapestry, making interactions across different languages a daily reality for individuals, businesses, and international entities. Whether for sealing international business deals, fostering diplomatic relations, conducting academic research, or simply connecting with friends and family abroad, the ability to communicate seamlessly and accurately across language barriers is paramount. The increasing need for seamless global interaction highlights a gap in current real-time translation, particularly for multi-party calls requiring voice preservation [1] and support for less-resourced languages. This project will develop a call translation app for at least three simultaneous users, enabling natural conversation in Hebrew, English, and Russian. Participants will hear others translated into their own language in near real-time, with the original speaker's vocal identity maintained. This will be achieved by integrating advanced AI: robust Automatic Speech Recognition (ASR), context-aware Neural Machine Translation (NMT) enhanced by Large Language Models (LLMs), and cutting-edge Text-to-Speech (TTS) with cross-lingual voice cloning. Speaker diarization will manage multi-user dynamics. A key focus is overcoming challenges in high-fidelity Hebrew voice-cloned output. The system will prioritize low latency and synchronization, aiming to revolutionize cross-lingual communication across business and personal spheres and set a new standard for accessibility.

What are the current existing solutions?

- **Google Meet's Live Translation:** Currently in beta and limited to specific language pairs (initially English-Spanish, with plans for others like German, Italian, and Portuguese), this feature aims for real-time audio-to-audio translation with voice preservation. However, support for Hebrew is not yet announced, and its broader availability and performance in diverse real-world scenarios are still to be seen.
- **Microsoft Translator:** Its multi-device conversation feature allows multiple users to communicate in their own languages, with translation delivered via text and standard TTS output. It supports a wide range of languages, including Hebrew and Russian for text and generic TTS, but it does not offer voice cloning.
- **Advanced Research Models:** Academic and research institutions are pushing the boundaries. For instance, Meta's SeamlessM4T [2] model demonstrates impressive multilingual and multimodal translation capabilities, including speech-to-speech translation with expression preservation for many languages. However, critical limitations exist, such as the current lack of Hebrew speech output in SeamlessM4T. These models often serve as research showcases rather than fully-fledged, commercially available multi-user applications.

Our project stands to significantly benefit a diverse range of users by dismantling communication barriers. From individuals seeking deeper personal connections across borders and businesses streamlining global operations, to educational institutions fostering international collaboration. Crucially, it offers enhanced communication capabilities for critical sectors like government and military operations, ultimately aiming to foster greater understanding and cooperation worldwide by making cross-lingual interaction more accessible and natural.

---

[1] Jia, Y., Macherey, W., Chen, Z., et al. (2022). *Translatotron 2: High-quality Direct Speech-to-Speech Translation with Voice Preservation.* arXiv preprint arXiv:2107.08661.
[2] Seamless Communication, et al. (2023). *SeamlessM4T: Massively Multilingual & Multimodal Machine Translation.* arXiv preprint arXiv:2308.11596.

2. Related Work

The development of real-time call translation systems, particularly those aiming for natural, voice-preserved, multilingual conversations, stands on the shoulders of significant advancements in several interconnected fields of artificial intelligence. This section provides an overview of these key areas and the current landscape of related solutions.

Key enabling technologies have matured considerably. Automatic Speech Recognition (ASR) systems are now capable of transcribing spoken language with high accuracy across numerous languages and accents, forming the crucial first step in converting speech to translatable text. Research continues to improve robustness in noisy environments and for low-resource languages, which is pertinent for languages like Hebrew with its unique phonetic and morphological characteristics, and for ensuring clarity in diverse conversational settings involving English and Russian.

Following ASR, Machine Translation (MT), particularly Neural Machine Translation (NMT) augmented by Large Language Models (LLMs), has revolutionized the quality, fluency, and contextual understanding of translated text. While high-resource language pairs achieve remarkable performance, achieving consistently low-latency, high-quality translations for conversational speech, especially for pairs like Hebrew-Russian or maintaining deep context in dynamic dialogues, remains an active area of research and development. Efforts are ongoing to create more comprehensive benchmarks and models specifically for languages such as Hebrew to address these gaps.

To ensure the clarity of input for ASR, AI-powered Noise Suppression technologies have become highly effective. These systems can intelligently distinguish speech from a wide variety of background noises, significantly improving the quality of the audio signal processed by the ASR and, consequently, the entire translation pipeline.

A critical component for a natural conversational experience is Voice Conversion and Cloning. This technology aims to synthesize translated speech not in a generic voice, but by preserving the original speaker's unique vocal identity, including tone and intonation. Advances in zero-shot voice cloning allow for real-time application with minimal speaker-specific data. While systems from major research entities and commercial providers demonstrate impressive capabilities for some languages, including Russian, achieving high-fidelity, emotionally resonant, and artifact-free cross-lingual voice cloning for all target languages, particularly for Hebrew, is an ongoing challenge and a key focus for creating truly immersive experiences.

While individual component technologies have advanced, Integrated Systems and Multi-Party Communication solutions that seamlessly combine these elements for real-time, multi-user call translation are still evolving. Current

commercial applications from major tech companies like Google (with its Meet live translation beta, initially limited in language pairs) and Microsoft (with its Translator supporting multi-device conversations using generic TTS) offer valuable functionalities but often do not provide the full suite of features envisioned, such as high-fidelity voice cloning for specific languages like Hebrew in a multi-user context. Academic research explores various architectures, including end-to-end Speech-to-Speech Translation (S2ST) models like Meta's SeamlessM4T (which supports Russian speech output but not Hebrew), aiming to reduce latency and improve prosody. However, challenges related to speaker diarization in multi-speaker scenarios, maintaining robust context across turns and languages, and managing the synchronization and latency of multiple audio streams persist, especially for systems involving three or more participants.

Our project aims to synthesize these advancements to create a real-time translation application specifically for Hebrew, English, and Russian users. By focusing on robust ASR, context-aware MT, and striving for natural voice cloning within an architecture initially designed for two-party communication (with potential for future expansion), we seek to address existing limitations in voice preservation, contextual accuracy, and specific language pair support. A more in-depth analysis of specific tools, algorithms, and performance metrics will be presented in the subsequent Literature Review.

3. Background

Global businesses, education and diplomacy increasingly demand seamless multilingual conversation tools. Cross-border calls, conferences and remote work require participants to communicate in real time despite language differences. Current consumer solutions largely handle one-to-one translation, or produce monolingual captions. There is a gap in robust multi-way conversation tools, especially for less-resourced languages like Hebrew and complex multi-speaker scenarios. Hebrew, for example and other less spoken languages lack the abundant training data of major languages and often suffer lower ASR/MT performance.

3.1.Speech-To-Text(STT)/Automatic-Speech-Recognition(ASR)

The first step in the translation pipeline is converting audio into text using an Automatic Speech Recognition (ASR) system. Recent advances in open-source models, such as OpenAI’s Whisper [3] and its optimized variants like Faster-Whisper, have significantly improved the accuracy and accessibility of multilingual ASR. These models support dozens of languages, including Hebrew, Russian, and English, and can operate in real-time on modern GPUs.

For low-latency applications, streaming ASR [4] is essential. Whisper-based models can be adapted for partial transcription while the speaker is still

talking, reducing end-to-end delay. However, Hebrew remains more challenging due to limited annotated training data and linguistic complexity, such as the lack of vowel markings.

Running ASR locally ensures better privacy, avoiding the need to transmit raw audio over the internet. On powerful desktop GPUs, ASR can process faster than real-time, but CPU-only systems may struggle with latency. The choice of ASR model size (small, medium, large) involves trade-offs between performance, accuracy, and hardware requirements.

---

[3] Radford, A., et al. (2022). *Robust Speech Recognition via Large-Scale Weak Supervision.* arXiv preprint arXiv:2212.04356.
[4] Pratap, V., et al. (2019). *Streaming End-to-End Speech Recognition for Mobile Devices.* arXiv preprint arXiv:1811.06621.

3.2. NMT (Neural Machine Translation)

Once speech is transcribed, an NMT system translates the text. Modern NMT leverages large multilingual models or context-aware LLMs. Research models like Meta's NLLB-200 [5] or M2M-100 [6] cover hundreds of languages and can be fine-tuned for speech corpora. Context-awareness as handling dialogue flow is emerging: new translation APIs and LLM chat frameworks (e.g. GPT-4/ChatGPT using interpreters) can incorporate conversation history to improve coherence. For real-time use, streaming/segmental NMT is key: many systems use "prefix-to-prefix" translation (translating as speech arrives) to cut latency. While fully streaming NMT is an active research area, practical systems often wait for sentence boundaries or use fixed-chunk translation. For our three languages, published benchmarks (BLEU scores, etc.) indicate all pairs can reach high quality on clean text, but mismatches (e.g. Hebrew orthographic ambiguity) can still cause errors. Using sub-sentence units or phrase-based translation may help in live calls, combined with post-editing by the TTS step.

---

[5] Team META AI. (2022). *No Language Left Behind: Scaling NMT to 200 Languages.* arXiv preprint arXiv:2212.09811.
[6] Fan, A., et al. (2020). *Beyond English-Centric Multilingual Machine Translation (M2M-100).* arXiv preprint arXiv:2010.11125.

3.3.1 Cloud translation

Services like DeepL, Google Translate, or Azure Cognitive Translator offer high accuracy and many languages. They typically take on the order of 100–300 ms per sentence. DeepL's API even has a "latency optimized" mode for faster results. For example, Azure's live speech translation has been benchmarked at sub-5-second latency for an entire spoken utterance. The trade-offs are obvious: cloud APIs yield excellent quality and up-to-date models, but require Internet connectivity and incur network delay and usage cost. They also send user text (or audio) to third-party servers, which can raise privacy concerns.

3.3.2 Local translation

Running an open-source translator locally avoids network delays and privacy issues. However, this approach poses several challenges. Running modern translation models requires substantial computational resources, especially for real-time performance. While lightweight or quantized models exist, they may offer limited language coverage or reduced accuracy. Additionally, translation speed can be slower compared to optimized cloud APIs, particularly for long or complex processes.

3.4. Voice Cloning

Synthesizing natural, expressive voices might help the output become more user friendly. State-of-art TTS pipelines use encoder-decoder models (e.g. Tacotron 2 / FastSpeech) paired with neural vocoders (WaveNet, HiFi-GAN, etc.) to produce realistic audio. Zero-shot voice cloning models [7] (like Meta's *VALL-E* or open-source variants) can mimic a speaker's timbre with only a few seconds of reference audio. For Hebrew especially, the challenge is limited training data: fewer high-quality, neutral Hebrew voice corpora exist, and Hebrew's prosody patterns differ from Romance languages.

3.5. Speaker Diarization

In a multi-user call, correctly attributing spoken segments to the right participant is essential. Speaker diarization models can separate mixed audio into speaker-labeled segments. Many open-source toolkits (like Pyannote [8] or SpeechBrain's EEND [9]) achieve high accuracy on close-talk or meeting-style audio. For real-time calls, one approach is to perform ASR on each audio stream separately (if each user's audio is on a separate channel) or use streaming diarization to tag speakers on a single mix.

---

[7] Wang, C., et al. (2023). *Neural Codec Language Models are Zero-Shot Text-to-Speech Synthesizers (VALL-E).* arXiv preprint arXiv:2301.02111.
[8] Bredin, H., et al. (2019). *pyannote.audio: Neural Building Blocks for Speaker Diarization.* arXiv preprint arXiv:1911.01255.
[9] Fujita, Y., et al. (2019). *End-to-End Neural Speaker Diarization with Self-Attention.* arXiv preprint arXiv:1909.06247.

4. Expected Achievements

Our project aims to create a mobile app that offers near real-time voice translation, making it easy for users to talk across different languages with very little delay. We know that achieving truly instant translation has its challenges because of all the complex technologies involved. However, our app will work to make these processes as efficient as possible, so conversations feel natural, smooth, and effortless.

4.1 Outcomes

The "Real-Time Call Translation" app will turn advanced multilingual speech-to-speech technology into something as simple and reliable as using your regular phone dialer. It will feature a clean, high-quality design that guides users through every step of a call. We are building it with accessibility in mind, so people of all technical backgrounds can use it easily, whether for casual chats or professional meetings.

Beyond the look and feel, the app will cleverly combine fast speech-to-text (ASR), smart machine translation, and voice cloning to deliver conversations that happen almost instantly. This means you'll hear others translated into your language, with their original voice characteristics preserved (for supported languages), removing language barriers and making communication more personal.

The user journey will start with a simple setup, where you pick your spoken and target languages, and then a quick voice enrollment step that lets you hear how voice cloning will sound. A contacts screen will show your friends and colleagues, noting which language they will hear. During a call, the screen will display live captions, show which language each person is speaking, and offer key controls like mute, hang-up, and language switching. All parts of the app will be built using responsive technology, ensuring a consistent experience on both Android and iOS phones.

4.2 Unique Features

4.2.1 Speech-To-Speech Translation

At the heart of the application is a fully integrated real-time speech-to-speech translation pipeline. This system captures the spoken input from users, transcribes it via Automatic Speech Recognition (ASR), translates the resulting text into the target language using a context-aware Neural Machine Translation (NMT) model, and finally converts the translation into natural-sounding speech using a Text-to-Speech (TTS) engine. To minimize perceptible delays and ensure conversational flow, the system will utilize stream-based processing, where audio is segmented and processed in parallel.

4.2.2 Voice Personalisation

To preserve speaker identity across linguistic barriers, the system incorporates advanced voice cloning technology. By utilizing a zero-shot model, a user's unique vocal characteristics including prosodic features such as timbre, intonation, and rhythm can be captured from a brief audio sample. This registered vocal profile is then used to synthesize the TTS output in the target language. The result is a translated audio stream that retains the original speaker's identifiable voice. This feature is critical for maintaining conversational cohesion and speaker attribution in multi-participant scenarios, thereby reducing the cognitive load required to track dialogue.

4.2.3 Multi-Participant Stream Management

Engineered for collaboration, the system excels in complex, multi-participant environments. Using real-time speaker diarization and voice activity detection, the platform accurately identifies and isolates each speaker's audio stream. It then delivers a personalized audio mix to each listener, where all other participants are translated into their preferred language while retaining their unique cloned voices. This ensures clarity and prevents cross-talk confusion in dynamic scenarios like international team meetings or group support calls.

4.2.4 Language Preference Configuration

To enhance efficiency and reduce translation latency, users will be prompted to predefine both their spoken language and the language in which they wish to receive translations. By bypassing the need for real-time language detection or language guessing algorithms, the application can optimize the translation pipeline and deliver faster, more accurate results. Furthermore, this feature enables language-aware speaker management in multi-user environments and simplifies the user interface by avoiding mid-call language configuration.

4.2.5 Unified and Intuitive Cross-Platform

Built on the Flutter framework, our application offers a consistent, responsive, and accessible interface across mobile, web, and desktop platforms. The design prioritizes clarity and control, providing users with at-a-glance information through clean visual cues: real-time indicators show who is speaking, what language is being processed, and the live status of the translation. This minimalist approach ensures users can manage complex multilingual conversations with minimal cognitive load.

4.3 Key Development Phases

Our Real-Time Call Translation app will be developed through a series of distinct phases, building upon the research and findings detailed in the preceding sections. This structured approach will ensure a robust and functional product, leading to the seamless multilingual communication experience we envision.

The development will progress through the following key phases:

1. **User Interface (UI) and Database Setup:** We will Establish the Flutter-based UI and Firebase for user data and voice sample storage, forming the app's foundation..
2. **Basic Server Setup:** We will set up the central server to handle call management and real-time audio streaming connections.
3. **Speech-to-Text (ASR) Integration:** We will integrate local Fast Whisper for converting incoming audio streams into text.
4. **Translation (NMT) Integration:** Connect the server to the Google Translate API for accurate text translation. We will also explore the potential for integrating GPT-4 for enhanced contextual understanding in future iterations.
5. **Text-to-Speech (TTS) and Initial Voice Synthesis Integration:** We will integrate Coqui xTTS for this purpose, focusing on generating natural-sounding audio.
6. **Multi-Participant Handling:** Develop features to identify and separate individual speakers in multi-party calls.
7. **Noise Suppression Integration:** We will implement techniques to enhance audio clarity before ASR processing.
8. **Voice Cloning and Personalization:** We will refine and integrate voice cloning using Coqui xTTS to preserve speaker identity in translated speech.
9. **Iterative Testing and Refinement:** Continuously test components and the integrated system, using feedback for optimization.
10. **Cloud Deployment:** Deploy server-side components to a scalable cloud environment (e.g., GCP) for reliability and global access.

4.4 Criteria For Success

The success of the app will be evaluated based on several key criteria:

1. **Real-time flow:** Conversations should feel natural, with translated speech arriving quickly enough that participants can speak without awkward pauses.

2. **Clear understanding:** Automatic speech recognition, translation, and synthesized output must be accurate enough that meaning is rarely lost or misinterpreted.
3. **Speaker identity:** The voice-cloning feature should preserve each participant's vocal character so listeners can instantly recognise who is talking, even after translation.

4. **Smooth group calls:** The system must manage several simultaneous speakers, keep their voices distinct, and deliver the right language mix to every participant.

5. **User friendliness:** Non-technical users should be able to install the app, set language preferences, and conduct a multilingual call without training or confusion.

6. **Reliability:** Calls should run to completion without crashes, noticeable audio glitches, or severe drops in quality, even on mid-range mobile devices.

5. The Process

5.1 Methodology And Iteration

From day one we treated the project as an experiment in rapid, evidence-driven iteration. We began with a broad "technology census," listing every promising library or service that could cover ASR, translation, or speech synthesis. Meta's SeamlessM4T served as our academic baseline: it is a single model that claims to perform ASR, MT, and TTS with voice preservation. After downloading its research Docker image we discovered it required a full Linux stack and high-end GPUs, yet it still ran several minutes per utterance on our test clips. That result convinced us to pursue a modular architecture.

For the next steps, we looked closely at open-source voice-cloning tools such as Coqui xTTS, FreeVC, and SpeechT5-VC. For each, we checked how well it worked with many languages, its speed in real-time, and how easy it was to adjust. At the same time, we tested different ASR (speech-to-text) options. We recorded the same audio samples in English, Hebrew, and Russian, then

processed them using Whisper (with its various settings), Google Cloud Speech-to-Text, and Azure Speech Service. We found that Whisper-large, running on our computer, provided good Hebrew accuracy; Google's streaming tool was easy to use but caused a slight delay due to network travel.

For translation, we linked up Google Cloud Translation, Azure Translator, and a GPT-4 setup that retained conversation history. Simple sentences translated well with all three. However, we found GPT-4 to be a little better with slang and jokes. But for speed, Google was likely faster.

For text-to-speech, a 2023 paper on XTTS by Casanova et al. led us to try out Coqui's version. We also evaluated commercial voices like Amazon Polly, Google Wavenet, and ElevenLabs. Even though ElevenLabs offered impressive quality, it didn't support Hebrew at the time, and it charged by each letter, which made it risky for live calls.

During all this testing, we used a basic Python program. This program recorded audio, sent it to Whisper, passed the text to each translation tool, and then turned it into speech using gTTS. This first program helped us confirm that everything worked together and showed us where delays could happen early on. Every two weeks, we planned to try new parts – like Faster-Whisper, xTTS, or Google's tools for telling speakers apart – and ran our tests again. This way, the test results helped us decide what to work on next.

This project proceeded in several research and development phases and branching, each addressing different challenges we encountered:

5.2 Technology Exploration And Information Gathering

5.2.1 Research

We opened the project with a comprehensive scan of real-time speech-translation tech. The very first goal was simple: find tools that support *live, multi-user, multilingual* audio while preserving each speaker's voice. Accordingly, we dived into Meta's SeamlessM4T and SeamlessM4T-v2 arguably the most advanced end-to-end speech-to-speech systems on record. They promise direct speech-to-speech output, voice-identity retention, and contextual understanding in one neural pass. We also pulled in open-source voice-cloning libraries FreeVC, SpeechT5-VC, and Coqui TTS/xTTS because voice preservation was a must-have. Parallel to voice work we benchmarked ASR options, cloud and local: Google Cloud Speech-to-Text in streaming mode, plus Whisper and Faster-Whisper compiled for GPU. Finally, we stood up three machine-translation back-ends Google Translate API, Azure Cognitive Services Translator, and GPT-based translation via the ChatGPT API and hammered them with English↔Hebrew and English↔Russian sentences. That hands-on exploration gave us a concrete picture of what public tools can *really* do and where proprietary research builds still fall short.

5.2.2 Constraints And Challenges

SeamlessM4T's public release turned out to be *demo-oriented*: incomplete features, Linux-only, Docker images that swallow GPU VRAM, and on ordinary Windows or non-GPU machines impossible runtime. Even on a beefy virtualized Linux box it was far too slow for live dialogue. FreeVC, SpeechT5, and early xTTS builds either lacked full multilingual support or ran with high, seconds-long latency. Google Cloud Speech-to-Text streamed fine but network round-trips added hundreds of milliseconds; Whisper was wonderfully accurate yet absolutely needs a GPU for real-time. For NMT, Google's API was fastest and rock-solid on common language pairs, whereas GPT-4 needed careful prompt engineering and delivered extra delay. Those realities pushed us to a *modular cascade*: Whisper for ASR, the fastest cloud or local NMT in the middle, and a voice-cloning TTS stage we can swap once latency and quality line up with the budget.

5.3 Speech Recognition and Translation

5.3.1 Research

We then evaluated full ASR + MT pipelines in Hebrew, Russian, and English. Whisper (medium and large) and Google's streaming ASR were clocked on short audio chunks to see where throughput collapsed. For translation we let Google NMT race GPT-4. We also trial ran Whisper's *multitask* mode which can emit translations directly and toyed with a "conversation memory" prompt for GPT-4 so that longer chats wouldn't drift off topic. The takeaway was clear: with aggressive chunking, a cascaded Whisper Google pipeline beats the latency target.

5.3.2 Constraints and Challenges

The bottleneck is compounded error: one Whisper typo becomes an off-base translation. Mis-segmenting short utterances in Hebrew remains a sore spot. GPT-4 is contextually brilliant but its extra few hundred milliseconds keep it on the bench for now. At this stage the live system handles Whisper ASR locally, shows Hebrew text, but still lacks end-to-end speech-to-speech, proper speaker diarization, and multi-user mixing.

5.4 Neural Machine Translation (NMT)

5.4.1 Research

We compared three translators head-to-head. Google Cloud Translation API gave fast, reliable results across 100+ languages, routinely clearing 500 ms. GPT-4 (via ChatGPT) handled idioms and ambiguous phrases better, especially when we fed it running dialog history. Azure Translator matched Google's speed but sounded stiffer in Hebrew and its pricing model felt trickier. All tests concentrated on *streamable, low-latency* modes because conversation flow can't pause for batch translation.

5.4.2 Constraints And Challenges

Hebrew complicates life: no written vowels and heavy morphology mean literal MT can stumble. Google's engine held tone well; GPT-4 wanted extra prompt structure; Azure occasionally lost nuance. Given the need for speed, Google became the working default, while GPT-4 is earmarked for future iterations aimed at richer context or multi-turn disambiguation.

5.5 Speech-To-Text (STT)

5.5.1 Research

Automatic Speech Recognition (ASR) is the first and most critical step in the translation pipeline. We focused on finding solutions that could operate in real-time or near real-time and be flexible enough to run locally or on cloud infrastructure

We tested and compared:

- OpenAI Whisper / Faster-Whisper These models provided the best overall balance of accuracy and flexibility. Whisper is robust across many accents and languages, and supports streaming modes using partial results. Its open-source nature gave us full control over deployment, though it required GPU support for real-time performance. We used Whisper locally in our first working prototype.

- Google Speech-to-Text API Google's cloud ASR system supports streaming and works efficiently with good transcription accuracy, especially for English. It also supports Hebrew and Russian, but latency is slightly higher than Whisper in our setup. Privacy concerns due to cloud transmission were also noted.

- Azure Speech Services Also accurate and developer-friendly, but it required more tuning for Hebrew, and the delay was higher in our testing.

Our implemented system uses Whisper as the default STT engine, running locally on a GPU-enabled machine. It supports real-time transcription, allowing the app to begin processing the translation before the user has even finished speaking, which helps reduce end-to-end latency.

5.5.2 Constraints And Challenges

Whisper stumbles on very short or noisy clips in Hebrew and sometimes splits or merges phrases at awkward points. Google's cloud ASR carries slightly higher latency and raises privacy concerns because audio leaves the device. Azure required extra tweaking for Hebrew and still lagged behind Whisper in speed. We are experimenting with sliding windows and dynamic chunking to refine Whisper's segment boundaries and preserve sub-second responsiveness.

5.6 Voice Conversion And Cloning

5.6.1 Research

Voice identity is the signature feature, so we pursued two tracks.Voice Cloning via Multilingual TTS Models and Integration of working API's. First, Coqui xTTS [10]: a zero-shot multilingual TTS that supports more than 17 languages and allows voice cloning with just a few seconds of reference audio. This was the most promising tool for real-time speech generation in the original speaker's voice. Second, FreeVC [11] and SpeechT5-VC [12]: these take neutral TTS and shift it into a target speaker's timbre. We collected voice samples from test users (10–30 seconds each) and used them with xTTS to synthesize personalized speech. The results were partially successful:

- For English and Russian, the cloning worked well.
- For Hebrew, there were issues with pronunciation accuracy and fluency.
- Long sentences sometimes produced robotic or inconsistent speech.

5.6.2 Constraints And Challenges

Hebrew pronunciation remains shaky because training data are scarce; long sentences can sound robotic. FreeVC and SpeechT5 shift prosody convincingly but run far too slowly for a live call. Integrating any cloning step complicates synchronisation with speaker diarization, and both GPU memory use and inference time are steep. For now, voice cloning is disabled in the live path while we seek ways to shorten latency and enrich Hebrew voice datasets.

---

[10] Casanova, E., et al. (2024). *XTTS: A Massively Multilingual Zero-Shot Text-to-Speech Model.* arXiv preprint arXiv:2406.04904.

[11] Li, J., Tu, W., & Xiao, L. (2022). *FreeVC: Towards High-Quality Text-Free One-Shot Voice Conversion.* arXiv preprint arXiv:2210.15418.

[12] Ao, J., et al. (2022). *SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing.* arXiv preprint arXiv:2110.07205.

5.7 Summary Of Findings

We compiled a comparison of the key technologies evaluated and our decisions can be found in the table below:

Category	**Technologies Evaluated**	**Findings / Decision**
Speech Recognition (ASR)	OpenAI Whisper (local)	High accuracy for multiple languages, runs offline with GPU – chosen for prototype. Requires optimization for real-time.
	Google Cloud STT API	Accurate and supports streaming, but adds network latency and external dependency.
	Azure Speech Service	Accurate (needed tuning for Hebrew), slightly higher latency; similar trade-offs as Google.
Translation (NMT)	Google Translate API	Fast (usually <0.5s per sentence), high quality for EN↔HE/RU – **used** in current demo for consistency.
	OpenAI GPT-4 (ChatGPT API)	Context-aware and fluent, but slower and costlier. Good for improving nuance, not used in real-time pipeline yet.
	Azure Translator	Reliable speed comparable to Google, but Hebrew output slightly less natural in tests.

TTS & Voice Cloning	Coqui XTTS (multilingual)	Zero-shot voice cloning across 17+ languages. Most promising for an integrated solution; Hebrew quality needs improvement, but used in experiments.
	FreeVC / Microsoft SpeechT5	Voice conversion tools to apply voice to existing speech. Worked offline but too slow for real-time; complex two-step pipeline.
	OpenAI TTS (beta)	Very natural voice quality, but no voice cloning and limited language support; not directly usable for our needs yet.
All-in-one S2ST	Meta SeamlessM4T	End-to-end speech-to-speech model with voice preservation. Impressive demo, but not practical for us (incomplete features, extremely high resource requirement).

6. The product

6.1 Requirements

6.1.1 Table 1: Functional Requirements

ID	Description
1	The system enables user sign-up and login functionalities, including language selection.
2	The system allows users to record a voice sample during registration for voice cloning purposes.
3	The system captures and transcribes speech in real-time into text using Automatic Speech Recognition (ASR).
4	The system translates transcribed text accurately into multiple target languages in near real-time.
5	The system synthesizes translated text back into speech.
6	The system clones and maintains the original speaker's voice characteristics during speech synthesis.
7	The system supports simultaneous multi-party conversations with real-time multilingual translation.
8	The system utilizes speaker diarization to correctly attribute speech segments to the corresponding speaker.
9	The system performs real-time noise suppression to enhance audio clarity.
10	The system displays real-time captions of translated speech on users' devices.
12	The system maintains user preferences including selected languages and voice profiles.

6.1.2 Table 2: Non-Functional Requirements

ID	Description
1	The system maintains end-to-end translation latency under 2 seconds to support natural conversations.
2	The system provides accurate and contextually coherent translations across various languages.
3	While being in call each user will be provided with clear, real-time text captions for each participant's translated speech.
4	Each translated speech segment displays the name of the original speaker for easy identification.
5	The interface will be intuitive, requiring minimal user training, with clear navigation and straightforward controls.
6	Will handle multiple simultaneous conversations without significant performance degradation.
7	Personal user data and voice samples are stored and protected against unauthorized access.
8	The system supports cross-platform compatibility, running seamlessly on both Android and iOS platforms.

6.2 Architecture Overview

This system enables real-time voice communication between users speaking different languages. It performs live speech-to-text transcription, translation, and speech synthesis using AI-based modules. The result is a seamless multilingual conversation where each user hears translated speech in their own language, optionally using voice cloning and live caption display. The system architecture of our project follows a client-server approach leveraging cloud-based resources, optimized for low latency, scalability, and user-friendly interactions. The main components include:

6.2.1 Users And Flutter Call App

Users interact with the system through a mobile application built with Flutter. The app allows users to initiate and receive calls, captures microphone input, and handles the playback of translated audio. It also displays translated text captions when enabled. Communication with the backend is done through two channels: HTTP for call setup and user management, and WebSocket for real-time audio streaming and responses.

6.2.2 Call Management And WebSocket Connection

When a user initiates a call, the Flutter app sends an HTTP call request to the backend's Call Management Service. This service handles call signaling, manages session states, and opens a WebSocket connection between the client and the server. The persistent WebSocket connection allows continuous audio streaming and real-time response delivery without re-establishing connections.

6.2.3 Central Server And Translation Pipeline

The central server acts as the core processing hub of the system. It receives audio streams over WebSocket, processes them through a multi-step AI pipeline, and returns translated output.

1. The server forwards incoming audio to the Whisper ASR module, which transcribes the speech into text using a multilingual speech recognition model.
2. The transcribed text is then passed to a translation module, typically implemented using Google Translate API, to convert it into the recipient's preferred language.
3. The translated text is sent to a Text-to-Speech (TTS) and voice cloning module. If a voice sample is available for the speaker, Coqui XTTS is used to generate translated speech in the original speaker's voice. If not, OpenAI TTS is used as a base, followed by Coqui's voice conversion to approximate the speaker's voice.
4. The final translated audio is sent back to the recipient's Flutter app through the same WebSocket connection. Optional captions are also sent as text.

This process allows two users to communicate in different languages while maintaining voice identity and conversational flow.

6.2.4 User Management Service

User data, such as profile information, language preferences, and voice samples, is handled by a dedicated User Management Service. The service exposes RESTful APIs for creating and updating user profiles. Users register and manage their data through the Flutter app, which sends HTTP requests to this service. Data is stored securely and efficiently using Firebase.

6.2.5 Firebase Integration

Firebase is used as the primary cloud backend for user management. It provides authentication services, real-time or Firestore database support for structured user data, and cloud storage for audio samples. The User Management Service interacts with Firebase to store and retrieve user profiles and voice data. During TTS processing, the central server queries Firebase for voice sample locations.

6.2.6 End-To-End Call Flow

When a user initiates a call:

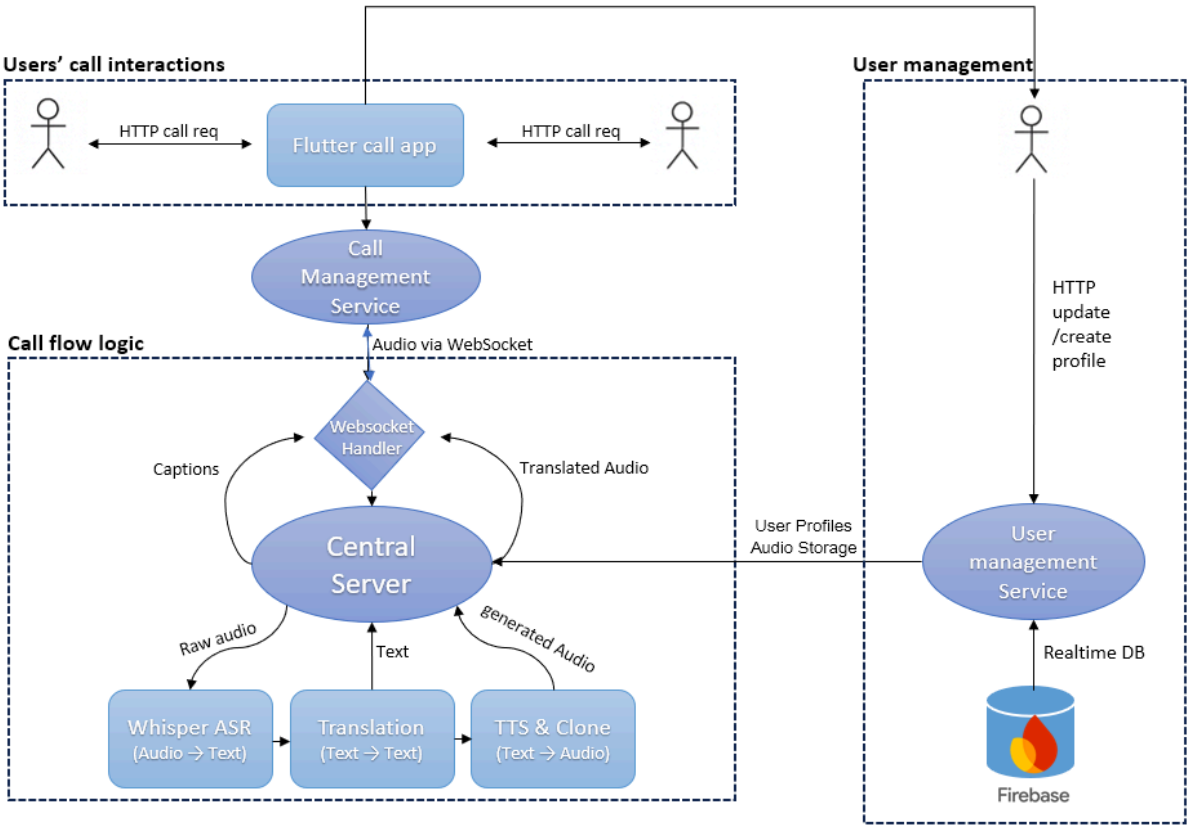
1. An HTTP request is sent to the Call Management Service to start the call session.
2. A WebSocket connection is established between the user and the server.
3. The app streams audio to the server.
4. The server transcribes, translates, and synthesizes the speech.
5. Translated audio and captions are streamed back to the recipient.
6. The same process occurs in reverse when the recipient speaks.

All audio and captions are sent through the WebSocket channel, enabling low-latency, real-time interaction between users.

6.2.7 Summary

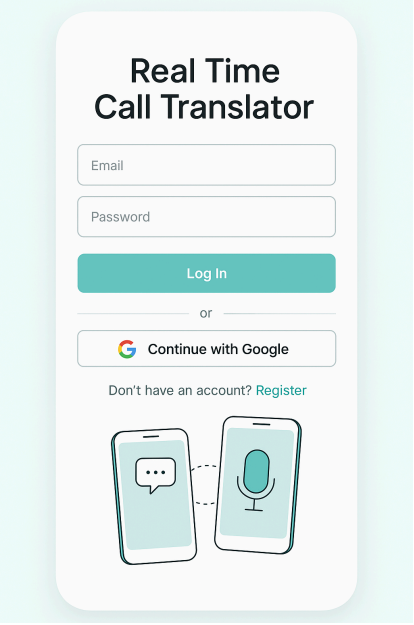
This architecture supports real-time multilingual conversation with high accuracy and natural voice reproduction. By combining Whisper ASR, cloud translation APIs, and Coqui-based TTS with Firebase for user management and storage, the system is designed to be efficient, scalable, and easy to extend. The use of a unified WebSocket channel for audio and caption streaming ensures smooth and responsive communication, making it ideal for real-world multilingual communication scenarios.

System Architecture: Real Time Translator



6.2.8 UI/UX

Main Screen



This screen is the **Login Screen** for the “Real Time Call Translator”
Here’s what you can do on this screen:

1. Login with Email and Password

    1.1. Enter your email address in the first field.
    1.2. Enter your password in the second field.
    1.3. Tap the “Log In” button to sign in with your credentials.

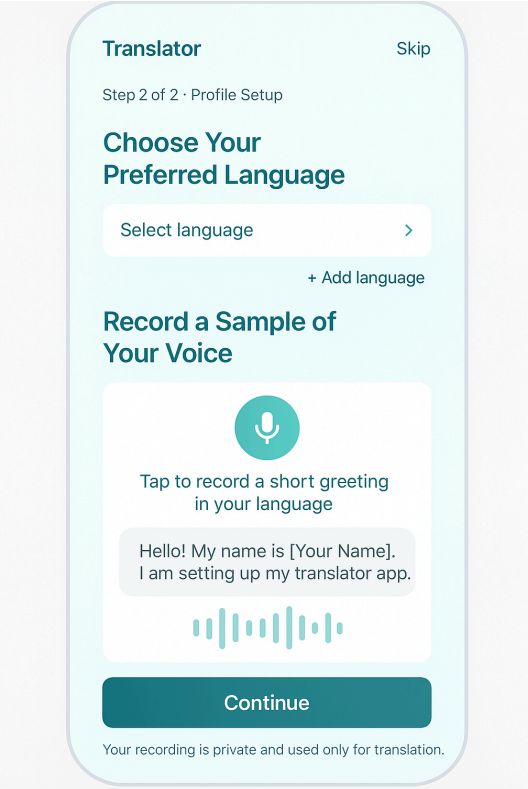
2. Log In with Google

    2.1. Instead of using email and password, you can tap the “Continue with Google” button to sign in with your Google account for quick access.

3. Register a New Account

    3.1. If you don’t have an account yet, you can tap “Register” (the blue link) to go to the registration or sign-up screen.

Profile Creation Screen



This screen is Step 2 of 2: Profile Setup for the Translator app. The functionality in this screen is:

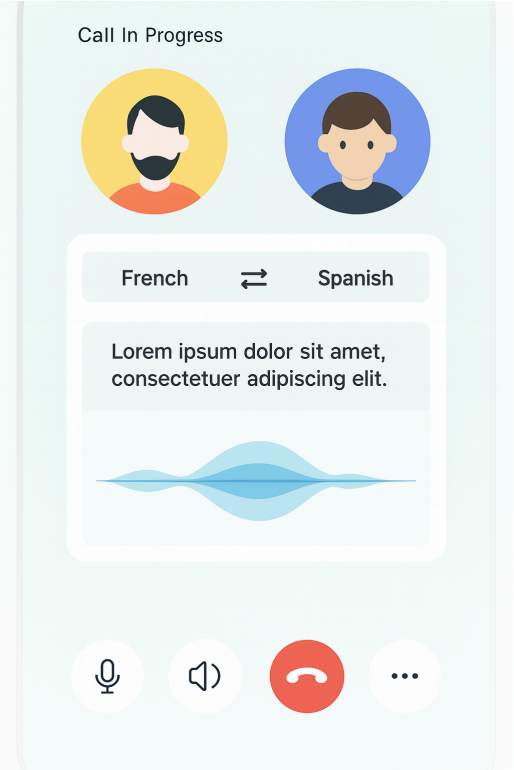
1. Choose Your Preferred Language

    1.1. Select your main language using the “Select language” dropdown menu.
    1.2. You can also tap “+ Add language” to include additional languages you want to use in the app.

2. Record a Sample of Your Voice

    2.1. Tap the microphone icon to record a short greeting in your language (for example: “Hello! My name is [Your Name]. I am setting up my translator app.”).
    2.2. This voice sample will help the app create a voice profile for more personalized translations (for example, for voice cloning or voice matching).

Call process Screen



1. View Call Participants
    1.1. See avatars of both people on the call at the top of the screen, each inside a colored circle.

2. Monitor Call Status
    2.1. The text “Call In Progress” at the top lets you know that you are currently in a live call.

3. Select Translation Languages
   3.1. The language selection bar in the middle shows the current languages being translated
   3.2. You can see which languages are active in the call for translation in each direction.

4. Read Live Captions
   4.1. box in the center displays the real-time translated captions of your conversation.

5. Visualize Audio Activity
   5.1. A waveform animation below the captions shows live audio activity or when someone is speaking.

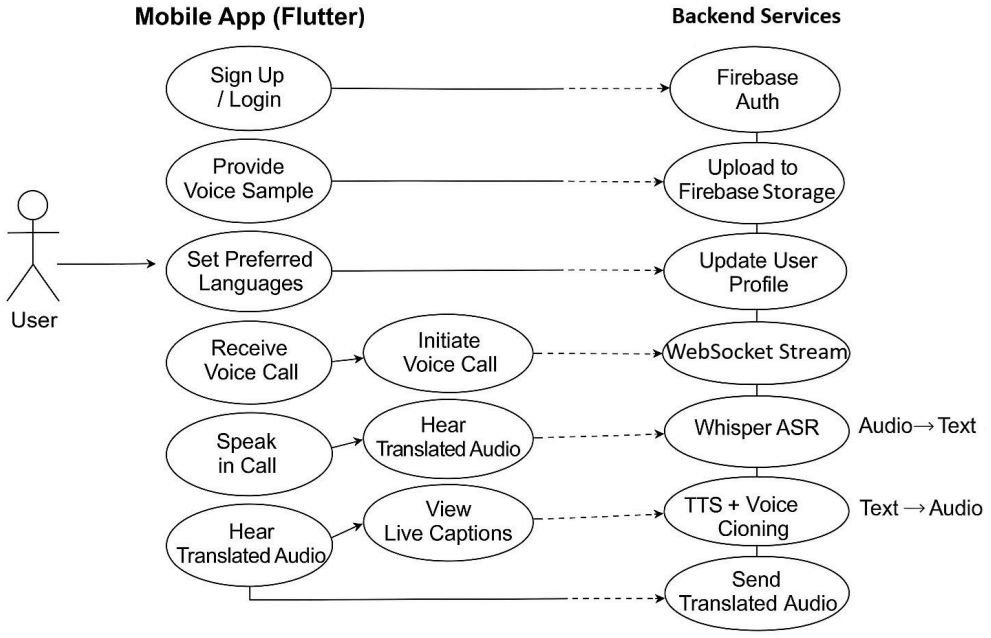
6. four main buttons:

   6.1. Microphone: Mute or unmute your microphone.
   6.2. Speaker: Turn the speaker on or off.
   6.3. Red End Call Button: Tap to hang up the call.
   6.4. More Options (three dots): Access additional call features or settings.

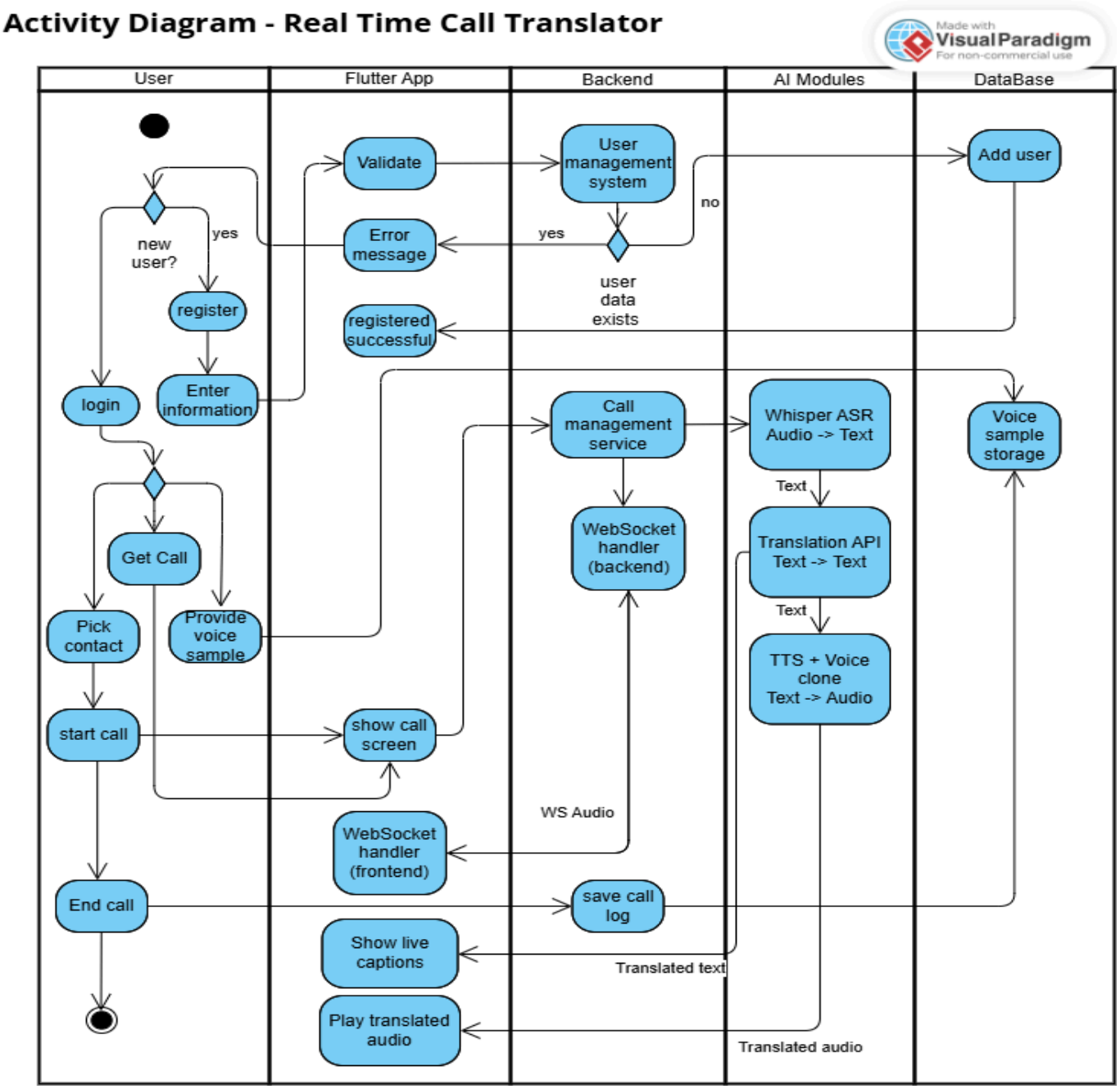
6.3 Diagrams

6.3.1 Use-Case Diagram

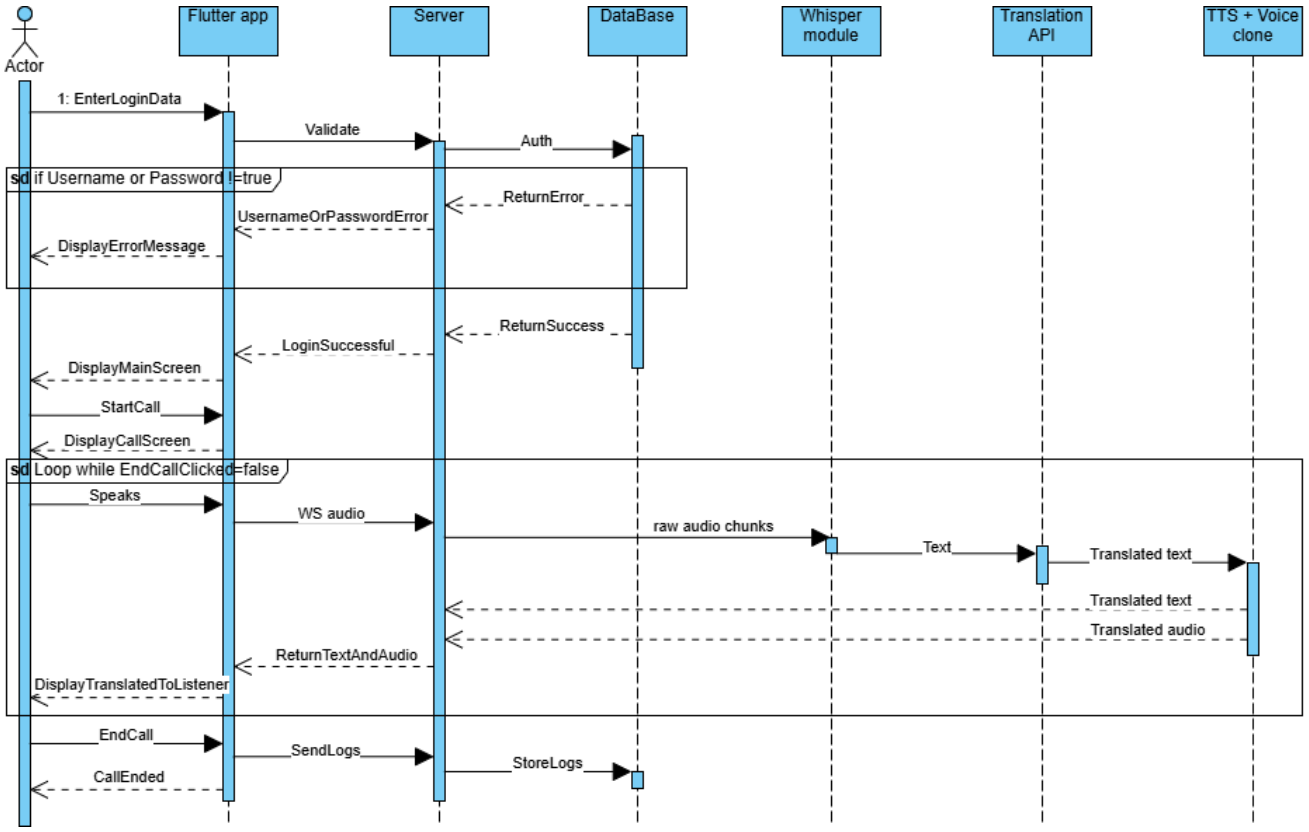
The diagram shows everything a User can do in the system and which backend services each action depends on.



6.3.2 Activity Diagram



6.3.3 Sequence Diagram



7. Verification And Evaluation

7.1 Evaluation

The effectiveness of the Real-Time Call Translation prototype will be judged chiefly on three fronts: conversational flow, intelligibility, and speaker authenticity. First, we will monitor how quickly the system converts live speech into an audible translation during multi-party calls, aiming for a delay short enough that participants can speak naturally without awkward pauses. Second, we will review the accuracy of both the automatic speech recognition and the machine-translation stages by comparing short recorded dialogues to their human-prepared references; the goal is to ensure that the intended meaning is preserved even under moderate background noise. Finally, we will rate how closely the synthesised voices resemble the original speakers and how natural those voices sound; translations must remain convincing and recognisable across different languages.

7.2 Verification

Test ID	Module	Tested Function	Expected Result
1	User Interface	Screen navigation and responsiveness	All screens change smoothly; controls react immediately to user actions.
2	User Interface	Language-preference workflow	Users can choose and save input/output languages without errors.
3	Data Storage	Profile read/write	New user profile is stored and re-loaded correctly after app restart.
4	ASR	Real-time partial transcription	Live captions appear while the speaker is talking and matching spoken words.
5	NMT	Context-aware translation	Translated text maintains meaning and fits the ongoing dialogue
6	TTS / Voice Cloning	Speaker-similarity synthesis	Output voice is recognisably similar to the original speaker and sounds natural.
7	Noise Suppression	Background-noise reduction	Speech remains clear when moderate ambient noise is present.
8	Speaker Diarization	Multi-speaker stream routing	Each utterance is attributed to the correct participant during a group call.
9	Call Orchestration	Latency under load	Conversation stays fluid when several users speak in short succession.

8. References

1. Jia, Y., Macherey, W., Chen, Z., et al. (2022). *Translatotron 2: High-quality Direct Speech-to-Speech Translation with Voice Preservation.* arXiv preprint arXiv:2107.08661.

2. Seamless Communication, et al. (2023). *SeamlessM4T: Massively Multilingual & Multimodal Machine Translation.* arXiv preprint arXiv:2308.11596.

3. Radford, A., et al. (2022). *Robust Speech Recognition via Large-Scale Weak Supervision.* arXiv preprint arXiv:2212.04356.

4. Pratap, V., et al. (2019). *Streaming End-to-End Speech Recognition for Mobile Devices.* arXiv preprint arXiv:1811.06621.

5. Team META AI. (2022). *No Language Left Behind: Scaling NMT to 200 Languages.* arXiv preprint arXiv:2212.09811.

6. Fan, A., et al. (2020). *Beyond English-Centric Multilingual Machine Translation (M2M-100).* arXiv preprint arXiv:2010.11125.

7. Wang, C., et al. (2023). *Neural Codec Language Models are Zero-Shot Text-to-Speech Synthesizers (VALL-E).* arXiv preprint arXiv:2301.02111.

8. Bredin, H., et al. (2019). *pyannote.audio: Neural Building Blocks for Speaker Diarization.* arXiv preprint arXiv:1911.01255.

9. Fujita, Y., et al. (2019). *End-to-End Neural Speaker Diarization with Self-Attention.* arXiv preprint arXiv:1909.06247.

10. Casanova, E., et al. (2024). *XTTS: A Massively Multilingual Zero-Shot Text-to-Speech Model.* arXiv preprint arXiv:2406.04904.

11. Li, J., Tu, W., & Xiao, L. (2022). *FreeVC: Towards High-Quality Text-Free One-Shot Voice Conversion.* arXiv preprint arXiv:2210.15418.

12. Ao, J., et al. (2022). *SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing.* arXiv preprint arXiv:2110.07205.