International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 11 | Nov 2026 www.irjet.net p-ISSN: 2395-0072

Voice Translation Application

Sure Mamatha[1], A.Gayathri[2], A.Mahima[3], B.Nipunitha[4], D.Anjana[5]

[12345]*Department of Computer Science and Engineering, Hyderabad Institute of Technology and Management, Hyderabad, Telangana State, India*

Abstract - *The Voice Translation Application is designed to help users communicate across language barriers by converting spoken input into translated text and, where supported, spoken output in a target language. The application follows a voice-processing workflow in which a user speaks into a device microphone, speech is converted into text, the text is translated through a language-processing API, and the translated result is displayed and optionally played as audio. The system is intended to support students, travellers, and users communicating with people who speak different languages. Its main components include voice input, speech-to-text processing, translation API integration, translated-text display, and text-to-speech output. API integration connects the user interface with external speech and translation services, while appropriate validation and error handling help manage unsupported languages, network failures, and unclear audio. The application demonstrates how speech technologies and translation services can be combined in one accessible interface. Further development can include additional languages, improved handling of noisy environments, conversation history, offline capabilities, and real-time two-way communication*

Keywords - Voice Translation, Speech Recognition, Speech-to-Text, Machine Translation, Text-to-Speech, API Integration, Multilingual Communication.

I. INTRODUCTION

Communication across languages can be difficult in education, travel, and everyday interactions. People may understand a language when reading it but find it challenging to speak or understand it in conversation. Manual translation requires users to type or copy sentences, which can interrupt the flow of communication. A voice-based translation application can reduce this effort by accepting spoken input and presenting the translated result in a language selected by the user. World Health Organization

Speech recognition, machine translation, and text-to-speech technologies make it possible to build applications that process spoken language through a sequence of digital services. A typical workflow captures audio, converts it into text, translates the text, and presents the result to the user. The quality of the result depends on audio clarity, language support, service availability, and network connectivity.

The proposed AI Voice Translation Application brings these steps together in a single user interface. It focuses on voice input, speech-to-text conversion, translation through an API, and displaying or vocalizing the translated output. The main contributions of the proposed system are:

The main contributions of the proposed system are:

1. A unified interface for voice-based translation.
2. Speech-to-text processing for converting spoken input into text.
3. Integration with a translation API to convert text between selected languages.
4. Display of the translated text for easy reading and verification.
5. Text-to-speech output where supported, allowing users to hear the translation.
6. Input validation and error messages for unsupported or unsuccessful requests.
7. A modular design that can be extended with more languages and conversation features.

II. RELATED WORK

A. Speech Recognition and Voice Interfaces

Speech-recognition systems convert audio signals into written text. Voice interfaces use this capability to allow users to interact with applications without typing every sentence. Recognition accuracy can vary with pronunciation, background noise, microphone quality, and the language selected. The proposed application uses speech-to-text as the first processing stage before translation.

B. Machine Translation Services

Machine translation services convert text from a source language into a target language. API-based services allow an application to send text and language settings to an external service and receive a translated result. The translation may not always preserve context, idioms, or specialized

terminology, so users should review important translations before relying on them.

C. Text-to-Speech and Multilingual Output

Text-to-speech technology converts written text into synthesized speech. When supported by the selected service and target language, it enables users to listen to a translation rather than only read it. Voice availability, pronunciation, and naturalness vary by language and service.World Health Organization

A voice translation workflow therefore benefits from combining speech recognition, translation, and speech synthesis while keeping each stage separate. This modular structure makes it easier to identify failures and replace or extend individual services.

D. Research Gap

Existing translation tools provide useful text and voice features, but users may still need to move between separate interfaces or repeat steps during a conversation. A simple, integrated workflow can make voice-based translation easier to access, especially for users who prefer speaking over typing.

The proposed application addresses this need by combining voice capture, speech-to-text conversion, translation API integration, and translated output in one workflow.

Voice-based translation requires multiple processing stages and dependable communication between the interface and external services. Common challenges include recognition errors, network delays, unsupported language pairs, and differences between spoken and written language. A clear workflow with input checks and error handling is therefore important.

The proposed system provides a centralized workflow containing:

- Voice input through a microphone
- Speech-to-text conversion
- Source and target language selection
- Translation API request and response handling
- Translated text display
- Text-to-speech playback where supported
- Error handling for failed requests
- A modular structure for future language and feature support

III. METHODOLOGY

A. System Design

The proposed application follows a modular client-service architecture. The user interface accepts the spoken input and language choices. A speech-recognition component converts audio into text, the translation API processes the text, and the translated result is returned to the interface. If text-to-speech is available, the translated text is also converted into audio. API integration manages requests and responses between the application and external services.

Design validation: The interface checks microphone permission and supported language pairs, and displays a clear retry message when an API request fails.

The overall architecture is:



Voice translation workflow:

User speaks
↓
Microphone audio
↓
Speech-to-Text Service
↓
Recognized Text
↓
Translation API
↓

Translated Text
↓
Text-to-Speech Service
↓
Audio / Text Output

B. System Modules

1. Voice Input and Language Selection

The input module allows users to provide speech through the device microphone and select the source and target languages. The application should check microphone permission and confirm that the selected languages are supported before processing the request.

2. Speech-to-Text

The speech-to-text module converts the captured audio into text using a supported speech-recognition service or browser capability. The recognized text is passed to the translation stage. Accuracy depends on the audio quality, pronunciation, and selected language.

3. Translation

The translation module sends the recognized text and language settings to a translation API.

- Source language
- Target language
- Recognized text
- API request status
- Translated text

The application receives the translation response and displays the result to the user. The interface should handle empty input, unsupported language pairs, and service errors.

4. Text-to-Speech

Where supported, the text-to-speech module reads the translated text aloud so the user can hear the result in the target language.

5. Results Display

The results module displays the recognized source text and translated text clearly so users can check the result before using it in conversation.

6. API Integration

The API integration module connects the application to speech-recognition and translation services. It manages request data, response parsing, service errors, and language settings.

7. Error Handling

The application informs users when microphone access is unavailable, speech cannot be recognized, the selected language is unsupported, or a service request fails.

8. Language Support

The application is designed to support translation between language pairs available through the integrated services. The list of supported languages depends on the selected APIs.

9. Translation Workflow

The system processes voice input through a sequence of stages. The general workflow is:

User Speech → Speech-to-Text → Translation API → Translated Text → Text-to-Speech / Display.

The actual languages, voice options, response time, and accuracy depend on the connected service and network conditions.

IV. IMPLEMENTATION

A. Frontend

The frontend provides the interface for microphone input, language selection, request status, recognized text, and translated output. Components can be organized by function so that input, processing feedback, and results remain easy to understand.

Responsive styling can be used to keep the interface usable across supported screen sizes.

B. Backend

The backend or API integration layer coordinates requests to external speech and translation services when required. It validates inputs, sends the required language and text parameters, processes responses, and returns results to the frontend. API credentials should be stored securely rather than exposed in client-side code.

A translation request conceptually includes:

POST /api/translate

for submitting recognized text and language settings. The response returns the translated text and status information.

GET /api/languages

for retrieving the supported language list, if this endpoint is implemented.

C. Database

The application may store user preferences or translation history in a database if these features are included. If history storage is not implemented, translation can be processed without persisting the text. Any stored voice or text data should be handled with appropriate privacy and security safeguards.

A translation-history record, if implemented, can contain:

sourceLanguage
targetLanguage
recognizedText
translatedText
requestStatus
createdAt
serviceName
responseTime

A user or session identifier can associate saved translation records with the appropriate session when history functionality is implemented.

V. RESULTS

The prototype workflow should be evaluated using representative speech input, language selection, translation requests, output display, and audio playback where supported. Testing should also cover empty input, microphone-permission denial, unsupported languages, network failures, and unsuccessful API responses.

TABLE I FUNCTIONAL TESTING OF THE PROPOSED SYSTEM

Module	Test Function	Result
Voice Input	Capture speech input	Verify in prototype
Speech-to-Text	Convert speech to text	Verify in prototype
Language Selection	Choose source/target languages	Verify in prototype
Translation API	Submit text for translation	Verify in prototype
Results Display	Show translated text	Verify in prototype
Text-to-Speech	Play translated output	If supported
Error Handling	Handle failed/empty requests	Verify in prototype
API Integration	Process request and response	Verify in prototype
Language Support	Check available languages	Verify in prototype
Network Handling	Display service/network errors	Verify in prototype
End-to-End Flow	Voice to translated output	Verify in prototype

Table I summarizes the core functional checks. Successful results should be recorded only for features that were actually run and verified in the submitted prototype. Translation quality and response time can vary according to the language pair, audio quality, and external service.

The basic end-to-end workflow is:

Voice Input
↓
Speech-to-Text
↓
Translation API
↓
Translated Text
↓
Text / Audio Output

The application depends on the capabilities and availability of the connected speech and translation services. The interface should clearly communicate processing status and provide a way to retry when a request fails. Users should review translations in important or sensitive conversations because automated translation can contain errors.

VI. DISCUSSION

The proposed application combines voice capture, speech recognition, translation, and translated output in one workflow. This can reduce the need to type text manually

and can make translation more convenient for users communicating across language barriers.

The quality of the output depends on the selected services, supported languages, microphone quality, pronunciation, background noise, and network connectivity. Speech recognition errors can affect the translated result, so showing the recognized text helps users identify mistakes. The system should also handle API failures without losing the user's selected language settings.

The application is intended as a communication aid and should not be treated as a guarantee of exact translation. Users should verify important legal, medical, or safety-related messages with a qualified person or trusted translation source.World Health Organization

VII. CONCLUSION

The AI Voice Translation Application provides a unified workflow for converting spoken input into translated text and, where supported, spoken output. Its main stages include voice input, speech-to-text conversion, translation API integration, and presentation of the translated result.

The modular design illustrates how speech and translation services can be connected through API requests and responses. A user-friendly interface, language selection, status feedback, and error handling are important for practical use.

Future work can include expanding language coverage, improving performance in noisy environments, adding conversation history with privacy controls, enabling offline translation where feasible, supporting two-way conversational mode, and evaluating translation quality with a broader set of users and language pairs.

ACKNOWLEDGMENT

The successful completion of the AI Voice Translation Application would not have been possible without the guidance, support, and encouragement of many people.

We express our sincere gratitude to our project guide for valuable guidance, continuous support, and constructive suggestions throughout the development of this project.

We are grateful to the Head of the Department for providing the necessary facilities, resources, and encouragement required to carry out this project.

We would also like to thank the Principal and faculty members of our institution for their support and for providing a suitable academic environment for completing this work.

We extend our sincere thanks to our friends and classmates for their valuable suggestions, technical assistance, and encouragement during the development and testing of the system.

Finally, we express our heartfelt gratitude to our parents and family members for their constant support, motivation, and encouragement throughout the project.

We are thankful to everyone who directly or indirectly contributed to the development of the AI Voice Translation Application.

REFERENCES

[1] Google Cloud, “Speech-to-Text documentation,” Google Cloud documentation.World Health Organization

[2] Google Cloud, “Cloud Translation documentation,” Google Cloud documentation.World Health Organization

[3] MDN Web Docs, “Web Speech API,” Mozilla Developer Network.PubMed

[4] W3C, “Web Speech API Specification,” World Wide Web Consortium.PubMed

[5] Google, “ML Kit: Translation,” Google for Developers.PubMed

[6] Google, “Text-to-Speech documentation,” Google Cloud documentation.PubMed

[7] World Wide Web Consortium, “WebRTC: Real-Time Communication in Browsers,” W3C documentation.PubMed Central (PMC)

[8] Microsoft, “Speech service documentation,” Microsoft Azure documentation.PubMed

[9] Amazon Web Services, “Amazon Transcribe documentation,” AWS documentation.PubMed

[10] Google for Developers, “Language support and translation services documentation.”World Health Organization

AUTHOR BIOGRAPHIES

First Author

B.Nipunitha is a student in the Department of Computer Science and Engineering at Hyderabad Institute of Technology and Management (HITAM), Hyderabad, Telangana. Their project contribution focuses on speech-to-text processing and API integration for the AI Voice Translation Application. Add the correct year of study and other approved biographical details before submission.