International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 11 | Nov 2025 www.irjet.net p-ISSN: 2395-0072

Speech-To-Text for Hearing Impaired

Asst. Prof. Madhuri Martis[1], Harshitha[2], Chaithanya S B[3], NaveenKumar[4], Deeksha P[5]

[1]Assistant Professor, Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology, Davangere, Karnataka, India.

[2,3,4,5]Bachelor of Engineering, Information Science and Engineering, Bapuji Institute of Engineering and Technology, Davangere, affiliated to VTU Belagavi, Karnataka, India.

Abstract - *This project presents a multilingual and emotion-aware Speech-to-Text (STT) system aimed at improving communication accessibility for hearing-impaired users. The system accepts English audio input, performs automatic speech recognition, and enhances the transcript through named-entity highlighting, part-of-speech emphasis, and keyword-based emotion detection. Additionally, it supports translation of the generated text into regional languages such as Kannada and Telugu to improve inclusivity for native speakers. Implemented as a lightweight Flask web application, the system integrates ASR, NLP, and translation modules while offering features such as emoji tagging, user-friendly visualization, and PDF export. Experimental testing demonstrates reliable transcription accuracy for short audio inputs and efficient real-time processing, making the system suitable for educational, assistive, and everyday communication environments..*

Key Words: Speech-to-Text (STT), Automatic Speech Recognition (ASR), Natural Language Processing (NLP), Emotion Detection, Multilingual Translation, Kannada, Telugu, Accessibility, Hearing-Impaired Users, Flask Web Application, Emoji Tagging, PDF Generation.

1. INTRODUCTION

In recent years, advancements in Artificial Intelligence (AI), Automatic Speech Recognition (ASR), and Natural Language Processing (NLP) have significantly improved human–computer interaction. Speech-to-text technologies, in particular, have become essential for accessibility, communication support, and real-time information processing. However, despite progress in ASR and Neural Machine Translation (NMT), many existing systems still struggle with regional languages, emotional understanding, and real-world noise conditions. These limitations make current tools less effective for hearing-impaired users who rely heavily on accurate, expressive, and context-aware transcription.

This project focuses on developing a multilingual and emotion-aware Speech-to-Text (STT) system that converts English audio into text, highlights linguistic features, identifies emotional tone, and translates the transcript into Kannada and Telugu. The system integrates ASR for transcription, NLP techniques for entity recognition and emotion detection, and lightweight translation models for regional language support. By adding emoji cues and semantic highlighting, the system enhances readability and user comprehension, especially for individuals with hearing impairments.

Although current STT systems such as Google Live Transcribe, Microsoft Azure Speech Services, and Whisper provide high accuracy, they often depend on strong internet connectivity, lack customization, or fail to support low-resource languages effectively. Many tools also overlook user experience (UX) elements like adjustable text size, contextual feedback, and inclusive design, which are essential for accessibility-based applications.

This project addresses these challenges by designing a lightweight, web-based application built using Flask that provides fast processing, entity highlighting, emotion tagging, multilingual translation, and PDF export capabilities. The proposed system aims to deliver an accessible, accurate, and user-friendly platform that enhances communication for hearing-impaired users and broadens linguistic inclusivity.

1.1 System Architecture

The proposed Speech-to-Text system is designed to be lightweight, modular, and highly accessible for hearing-impaired users. Its architecture ensures accurate transcription, emotion detection, multilingual translation, and smooth user interaction. To guarantee efficient operation, the system integrates several coordinated layers and technologies.

1. Frontend Layer: The user interface is developed using HTML5, CSS3, JavaScript, and Bootstrap, providing a simple and responsive experience. It allows users to upload or record audio, view highlighted transcripts, select translation options, and download the final PDF output with ease.

2. Backend Layer: The backend is built using the Flask framework, which manages all user requests, processes audio files, handles NLP operations, and coordinates ASR, emotion detection, and translation modules. This layer controls the data flow between audio input, text processing, translation, and PDF generation.

3. Audio Processing & Speech Recognition Module: This module uses Python's SpeechRecognition library and Google's Web Speech API to convert uploaded .wav audio into English text. It ensures accurate Automatic Speech

Recognition (ASR) even under moderate background noise. It forms the core of the system's speech-to-text capability.

4. NLP & Emotion Detection Module: Using spaCy and keyword-based sentiment detection, the system analyzes the transcript for:

- named entities
- verbs and adjectives
- emotion-related keywords (happy, sad, angry, etc.)

Detected emotions are mapped to suitable emoji tags, improving clarity and expressiveness for hearing-impaired users.

5. Translation Layer: The translated output is generated using googletrans or Deep Translator, converting English text into Kannada or Telugu. This layer enables multilingual accessibility and enhances regional usability. It handles API request limits and provides fallback translation when needed.

6. PDF Generation Layer: Using FPDF/ReportLab, the system compiles the processed transcript—complete with highlights, emojis, and translation—into a downloadable PDF. This ensures portability and easy documentation for users.

7. Optional Data Handling Layer: Though the current system does not store user data permanently, it can be extended to use SQLite/MySQL for saving transcripts, translations, and logs if future enhancements require it.

1.2 Key Technologies used

- Frontend Interface Development: The user interface is built using HTML5, CSS3, JavaScript, and Bootstrap, providing a clean, responsive, and accessible layout. This enables users to easily upload audio files, view transcripts, select translation options, and download PDFs.
- Backend Framework: The system uses the Flask web framework to handle user requests, manage audio uploads, run ASR and NLP pipelines, and return processed results. Flask ensures lightweight, fast, and scalable backend operations.
- Speech Recognition Engine (ASR): Speech-to-text conversion is performed using Python's SpeechRecognition library integrated with Google Web Speech API, enabling accurate English transcription from .wav audio files.
- NLP Processing – spaCy/TextBlob: Natural language processing tasks such as tokenization, entity recognition, sentiment evaluation, and keyword-based emotion detection are handled using spaCy and TextBlob, making the transcript more expressive and context-aware.
- Multilingual Translation – Googletrans / Deep Translator:
  The system incorporates googletrans or deep-translator to translate English text into Kannada and Telugu, improving accessibility for regional language users.
- PDF Generation – FPDF / ReportLab**:** To provide downloadable transcripts, the system uses FPDF or ReportLab to generate professionally formatted PDF documents that include original text, translations, highlights, and emojis.
- Audio Handling Libraries: Tools such as pydub and wave are used for validating and preprocessing audio files, ensuring proper input format and smooth ASR processing.
- User Interface Enhancements: Features like emotion-based emoji tagging, highlighted entities, and modular HTML templates make the interface intuitive and user-friendly for hearing-impaired users.

2. FEATURES AND FUNCTIONALITY

The proposed Speech-to-Text system supports hearing-impaired users by combining speech recognition, emotion detection, and multilingual translation within a simple web-based design. Its core features focus on improving accessibility, clarity, and real-time communication support.

2.1 Speech Input and Transcription

- Users can upload or record English audio in .wav format through the web interface.
- The system converts speech into accurate text using Automatic Speech Recognition (ASR).
- It handles different accents and moderate background noise to ensure better transcription quality.

2.2 Emotion and Text Enhancement

- The system detects basic emotions such as happy, sad, angry, or neutral using NLP-based keyword analysis.
- Emoji tagging and text highlighting make the transcript more expressive and easier to understand for hearing-impaired users.
- Named Entity Recognition (NER) emphasizes key information like names, places, and dates for clearer context.

2.3 Multilingual Translation

- The transcribed English text can be translated into Kannada or Telugu.
- This feature improves accessibility for regional language speakers.
- Users can choose the preferred language based on personal or communication needs.

2.4 PDF Output and Documentation

- The system allows users to download the final transcript as a PDF file.
- The PDF includes the original text, emojis, highlights, and translated content.
- This enables users to save, share, or reuse the transcript for educational or communication purposes.

2.5 User-Friendly Web Interface

- A simple, clean, and responsive interface built with HTML, CSS, JavaScript, and Bootstrap.
- Users can easily navigate steps like audio upload, transcription, translation, and download.
- The design supports accessibility for users with limited technical experience.

2.6 Security and Privacy

- Uploaded audio files are processed temporarily and are not stored, ensuring user privacy.
- The system avoids long-term data storage unless extended for future enhancements.
- This approach protects sensitive speech information and maintains confidentiality.

3. Problem Statement

Despite major advancements in speech recognition and translation technologies, developing a lightweight, multilingual, and emotion-aware speech-to-text system remains challenging. Existing tools often struggle to accurately process regional Indian languages like Kannada and Telugu due to limited linguistic resources. Most current systems also lack effective emotion or context-aware features—such as semantic highlighting or emoji-based expression—that can help hearing-impaired users interpret tone and intent. In addition, translation services and APIs may face instability, request-size limits, and quota restrictions, affecting real-time performance. Therefore, there is a need for a robust web-based solution that can efficiently convert speech into text, identify emotional cues, support multilingual translation, and provide reliable output for accessible communication.

3.1 Existing System

Current Speech-to-Text tools like Google Live Transcribe, Microsoft Azure, and IBM Watson provide real-time transcription but depend heavily on internet connectivity and are not customized for hearing-impaired users. Most existing systems lack features such as emotion detection, visual cues, multilingual support for regional languages like Kannada and Telugu, and accessibility-focused interface design. Advanced models like Whisper offer better accuracy but require high computing resources, making them difficult for regular users to deploy. Overall, existing solutions are general-purpose and do not fully address the specific communication needs of hearing-impaired individuals.

3.2 Proposed System

The proposed system is a multilingual, emotion-aware Speech-to-Text (STT) web application designed specifically to support hearing-impaired users. It converts English speech into text using Automatic Speech Recognition and enhances the output through keyword highlighting, named entity recognition, and emoji-based emotion tagging. To improve accessibility, the system also provides translation into Kannada and Telugu, allowing users to view the transcript in their preferred regional language. Built using a lightweight Flask backend, the system enables quick processing, a simple user interface, and seamless interaction without complex hardware requirements. Users can upload or record audio, view the processed transcript, and download the final result as a PDF. The design focuses on accuracy, ease of use, and inclusive communication, offering a more tailored solution than existing generic STT tools.

4. System Requirements and Specifications

The system requires a combination of software and hardware components to enable audio input, speech-to-text conversion, emotion detection, translation, and PDF generation. The specifications are designed to ensure fast processing, accurate transcription, and a smooth user experience.

4.1 Functional Requirements

- Audio Input: Users must be able to record or upload .wav audio files through the web interface.
- Speech-to-Text Conversion: The system should accurately convert English speech into text using ASR.
- Emotion Detection: The system should identify basic emotions (happy, sad, angry, neutral) from the transcribed text.
- Emoji Representation: Detected emotions should be mapped to suitable emojis for better expression.
- Translation: Users should have the option to translate the transcript into Kannada or Telugu.
- Display Output: The processed text, emojis, and translation must be shown clearly on the web interface.
- PDF Download: Users must be able to download the transcript (with translation and emojis) in PDF format.

4.2 Non-Functional Requirements

- Performance: A 10-second audio clip should be processed in under 5 seconds.
- Reliability: The system should maintain high uptime and handle errors gracefully.
- Usability: The interface should be simple and accessible for users with no technical background.
- Security: Uploaded audio should not be stored permanently, ensuring privacy.
- Scalability: The system should support multiple users and be deployable on cloud platforms.
- Maintainability: Modular design should allow easy updates to ASR, NLP, or translation components.

4.3 Hardware Requirements

- Processor: Intel Core i5 / AMD Ryzen 5 or higher
- RAM: Minimum 8 GB (Recommended 16 GB)
- Storage: Minimum 256 GB SSD or 500 GB HDD
- Microphone: Built-in or external mic for audio input
- Internet Connection: Required for API-based ASR and translation

4.4 Software Requirements

- **Operating System:** Windows 10/11 or Ubuntu 22.04
- **Programming Language:** Python 3.8+
- **Framework:** Flask
- **Libraries:** SpeechRecognition, spaCy/TextBlob, googletrans, FPDF/ReportLab
- **Frontend:** HTML, CSS, JavaScript, Bootstrap
- **Browser:** Google Chrome / Firefox

5. System Design

5.1 System Architecture

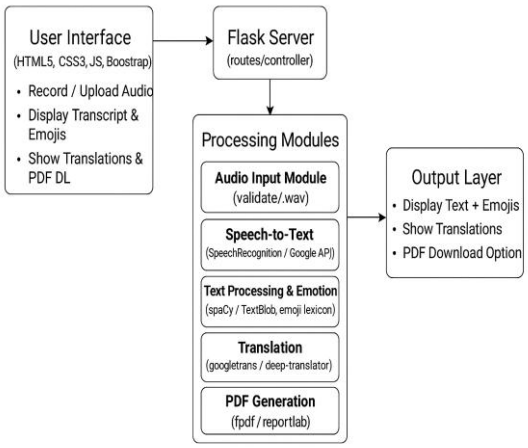

Fig 5.1 System Architecture of the Speech-tApplication

5.2 Use Case Diagram

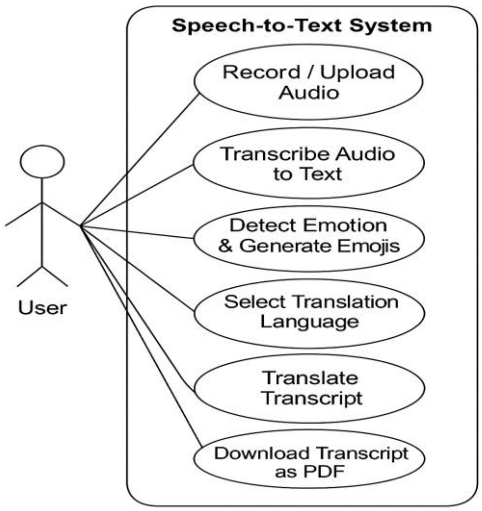

Fig 5.2 Use Case Diagram of the Speech-to-Text System

6. Testing

The goal of testing in the Speech-to-Text system is to verify how the different modules—such as the Audio Input Module, Speech Recognition Module, NLP & Emotion Detection Module, Translation Module, and PDF Generation Module—work together as a single, unified system. The objective is to ensure that each independently developed component transfers data correctly and produces accurate, real-time results for the user.

How Integration Testing Operates

Identify Integrated Modules:

- Modules such as audio upload, speech-to-text conversion, emotion detection, translation, and PDF export are selected for combined testing.
- Each module is evaluated to ensure it can successfully pass its output to the next stage.

Define Integration Workflows:

- The full workflow is tested step-by-step:
  Audio Input → Transcription → Emotion Detection → Translation → PDF Generation
- Scenarios like noisy audio, neutral speech, or translation requests are included to test real-world behavior.

Conduct Integration Tests:

- Test audio files are uploaded through the interface to simulate real user input.
- The system processes the audio through all modules, and the data flow is observed from ASR output to translation and final PDF.

Validate Data Flow:

- The output of one module (e.g., transcribed text) is checked as valid input for the next module (e.g., emotion tagging).

- Translated text is verified before being passed into the PDF generation stage to confirm seamless module interaction.

Debugging and Retesting:

- Any transcription errors, translation failures, or emoji mismatches are recorded.
- Corrections are applied, and the entire workflow is retested to confirm smooth performance and accurate output across all modules.

7. Results and Discussion

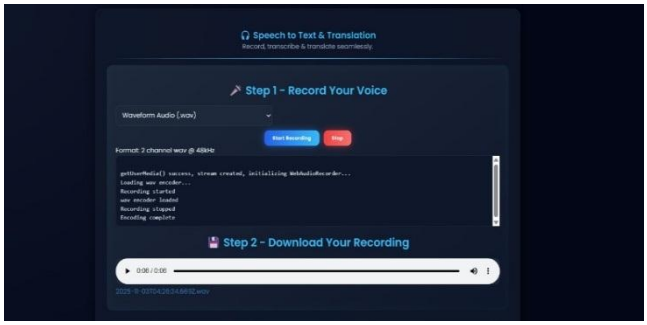

Fig -7.1: Audio Recording Screen(Step 1 – Record Voice)

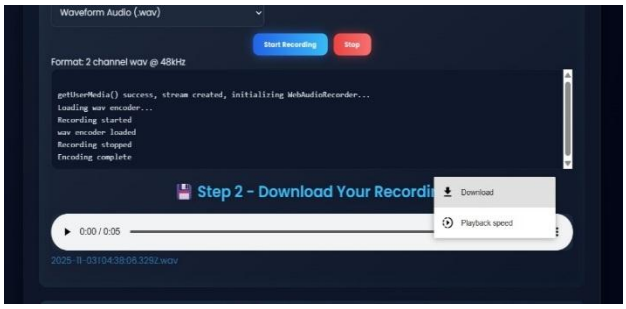

Fig – 7.2: Download Recording Option(Step 2 – Download)

Fig -7.3: English Transcription Output (Step 4 – Convert & Translate)

Fig -7.4: Kannada Translation Output (Step 4 – Convert & Translate)

8. CONCLUSION

The Speech-to-Text system successfully integrates speech recognition, emotion detection, and multilingual translation into a single, user-friendly web application designed to support hearing-impaired users. By converting English audio into text, highlighting key information, tagging emotions with emojis, and providing translation to Kannada and Telugu, the system enhances clarity, accessibility, and context in communication.

Overall, the project demonstrates that combining ASR, NLP, and translation technologies can bridge communication gaps and provide a more inclusive digital experience. With future enhancements—such as advanced ASR models, improved emotion recognition, offline capability, and broader language support—the system can evolve into a more powerful tool for education, assistive communication, and everyday interaction for hearing-impaired individuals. The lightweight Flask-based architecture enables fast processing, minimal resource usage, and easy deployment, while features like PDF export make the output practical for real-world use. Testing showed reliable transcription accuracy for short audio clips and efficient performance across modules.

REFERENCES

[1] M. Supriya, “Enhancing Neural Machine Translation Quality for Low-Resource Dravidian Languages,” Technologies, 2024. Available at: https://www.mdpi.com/

[2] S. M. George, “A Review on Speech Emotion Recognition: Recent Advances and Trends,” Journal of Neural Networks and Signal Processing, 2024. Available at: https://www.sciencedirect.com/

[3] K. Machová, “Detection of Emotion by Text Analysis Using Machine Learning,” Frontiers in Psychology, 2023. Available at: https://www.frontiersin.org/

[4] “LowResource KannadaSpeech Recognitio using Lattice RescoringandSpeechSynthesis,”IndianJournalofScienceand Technology.Availableat:https://www.indjst.org/

[5] Explosion AI, "spaCy — Industrial Strength NLP," 2024. Available at: https://spacy.io/

[6] W. Slam, "Frontier Research on Low-Resource Speech Recognition: Techniques and Challenges," Scientific Reports,2023. Available at: https://www.nature.com/srep/

[7] M. S. Fahad, "A Survey of Speech Emotion Recognition in NaturalEnvironments,"JournalofSpeechanSignalProcessing, 2021.Availableat:https://www.sciencedirect.com/

[8] H. Lian et al., "A Survey of Deep Learning-Based Multimodal Emotion Recognition," Entropy, 2023. Available at: https://www.mdpi.com/journal/entropy

[9] A. Bisht et al., "Neural Machine Translation for Low Resource Indian Languages," IETE Journal, 2022. Available at: https://ietejournal.org/

[10] CFILT, "Speech Emotion Recognition Survey," IIT Bombay, 2023. Available at: https://www.cfilt.iitb.ac.in/

[11]J.deLopeetal.,"AOngoingReviewofSpeechEmotionRecognition,"2023.Availableat:https://scholar.google.com/

[12] Y. A. Ejigu, "Large Scale Speech Recognition for Low Resource Languages," 2024. Available at: https://arxiv.org/

[13] P. Jain et al., "Performance Analysis of End-to-End ASR Models," EURASIP Journal on Audio, Speech, and Music Processing,2025.Availableat:https://asmp.eurasipjournals.springeropen.com/

[14 M. Nivaashini,"Deep Neural Machine Translation for IndianLanguages,"Wiley,2024.Availableat:https://www.wiley.com/

[15] Semantic Scholar, "Neural Machine Translation for Low-Resourced Indian Languages," 2020–2024. Available at: https://www.semanticscholar.org/