**BRIDGING THE SILENCE:PHONE CALLS FOR DEAF PEOPLE TO ATTEND AND COMMUNICATE**

**A SOCIALLY RELEVANT MINIPROJECT REPORT**

***Submitted by***

# VARSHITA SARWAN [211423104719]

**VIDUSHI GANESHIKA M[211423104730]**

***in partial fulfillment for the award of the degree***

***of***

# BACHELOR OF ENGINEERING

***in***

## COMPUTER SCIENCE AND ENGINEERING



**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna UniversityChennai)**

OCTOBER 2025

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

## BONAFIDE CERTIFICATE

Certified that this project report **“BRIDGING THE SILENCE:PHONE CALLS FOR DEAF PEOPLE TO ATTEND AND COMMUNICATE”** is the bonafide work of **VARSHITA SARWAN(211423104719),VIDUSHI GANESHIKA M(211423104730)**who carried out the project work under my supervision.

**Signature of the HOD with date Signature of the Supervisor with date**

**Dr L.JABASHEELA M.E.,Ph.D., Dr.V. SUBEDHA M.Tech.,Ph.D.,**

Professor and Head, Professor,

Department of Computer Science Department of Computer Science and Engineering, and Engineering,

Panimalar Engineering college Panimalar Engineering College, Chennai- 123 Chennai- 123

Submitted for the socially relevant miniproject Viva–Voce examination held on \_\_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

### DECLARATION BY THE STUDENT

We **VARSHITA SARWAN[211423104719],VIDUSHI GANESHIKA M [211423104730]** hereby declare that this project report titled **“BRIDGING THE SILENCE:PHONE CALLS FOR DEAF PEOPLE TO ATTEND AND COMMUNICATE”,** under the guidance of **Dr.V. SUBEDHA M.Tech.,Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**VARSHITA SARWAN [211423104719]**

**VIDSHI GANESHIKA.M [211423104730]**

# ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHIKUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHIKUMAR, B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.,** for his timelyconcern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr.L.JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank our Project Coordinator **MRS.S.SHARMILA, M.E., Ph.D.,** and our Project Guide **Dr.V.SUBEDHA, M.Tech., Ph.D.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

.

**VARSHITA SARWAN[211423104719]**

**VIDUSHI GANESHIKA M [211423104730]**

**ABSTRACT**

People with hearing and speech impairments face major barriers when engaging in phone calls, online meetings, or real-time conversations. These challenges extend beyond social interactions into education, employment, and emergencies. For instance, someone who cannot hear may struggle to follow a business meeting, while a person with a speech impairment may feel helpless when trying to describe a medical emergency. Such obstacles can lead to exclusion, isolation, and in critical cases, delays in receiving urgent help.

AI-based accessibility technologies are emerging as powerful solutions to bridge these gaps. Speech-to-text (STT) tools like OpenAI’s Whisper and Google Speech-to-Text can transcribe spoken language into written text in real time, enabling hearing-impaired individuals to read conversations instantly. This is particularly useful in classrooms, workplaces, and urgent phone calls. For people with speech impairments, text-to-speech (TTS) technology offers an equally valuable solution. By converting typed text into natural-sounding speech, TTS allows users to communicate more confidently and effectively with hearing individuals, going beyond the limits of text-only communication.

Further advancements include AI-driven sign language recognition using frameworks such as MediaPipe or TensorFlow. These systems can translate sign language into text or speech, while reverse translation through animated avatars can make conversations more natural for sign language users. When integrated into a unified platform, these technologies can provide seamless, adaptive communication whether through transcription, synthesized speech, or sign language. By harnessing AI, society can move toward a future where people with hearing and speech impairments are fully empowered to participate in education, work, daily life, and emergencies.

In addition to accessibility, AI solutions also promote inclusivity by fostering equal participation across diverse environments. In educational institutions, AI tools can ensure that students with hearing or speech impairments do not miss lectures or discussions, helping them to learn at the same pace as their peers. In workplaces, these tools can break down communication barriers, enabling employees to collaborate more effectively, contribute ideas without hesitation, and achieve career growth without being limited by their disabilities.

Moreover, the integration of these technologies into emergency response systems has life-saving potential. For example, a person with a speech impairment can use TTS to report emergencies instantly to authorities, while AI-enabled transcription tools can help first responders understand critical details in noisy or chaotic environments. In public service settings, such as hospitals, transport hubs, or government offices, these technologies ensure that people with impairments can access information and assistance without dependency on human intermediaries.

Looking ahead, the convergence of STT, TTS, and sign language recognition with wearable devices and mobile applications could make communication even more seamless. Glasses with real-time captioning, smartwatches with haptic alerts, or portable AI interpreters could transform how individuals with hearing and speech impairments interact with the world. As research and development continue, the vision is clear: building a society where technology erases communication barriers and fosters equal opportunity for all.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **NO** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | v |
|  | **LIST OF FIGURES** | ix |
|  | **LIST OF TABLE** | x |
| **1.** | **INTRODUCTION** | **1** |
|  | 1.1 Overview | 1 |
|  | 1.2 Problem Definition | 2 |
|  | 1.3 Literature Survey | 3 |
| **2.** | **SYSTEM ANALYSIS** | **5** |
|  | 2.1 Existing System | 4 |
|  | 2.2 Proposed System | 4 |
|  | 2.3Implementation Environment | 5 |
| **3.** | **SYSTEM DESIGN** | **7** |
|  | 3.1 UML Diagrams | 7 |
| **4.** | **SYSTEM ARCHITECTURE** | 9 |
|  | 4.1 Architecture Diagram | 9 |
|  | 4.2 Module Design Specification | 10 |
| **5.** | **SYSTEM IMPLEMENTATION** | **13** |
|  | 5.1 Backend Coding | 13 |

|  |  |  |
| --- | --- | --- |
| **6.** | **PERFORMANCE ANALYSIS** | **20** |
|  | 6.1 Result and Discussions | 20 |
| **7.** | **CONCLUSION** | 22 |
|  | 7.1 Conclusion | 22 |
|  | 7.2 Future enhancement | 23 |
| **8.** | **APPENDICES** | 24 |
|  | A1 SDG goals | 24 |
|  | A2 Sample Screenshots | 25 |
|  | A3 Paper Publication | 29 |
|  | A4 Plagiarism report | 30 |
| **9.** | **REFERENCES** | **31** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG NO.** | **FIGURE DESCRIPTION** | **PAGE NO** |
| 3.1.1 | System architecture diagram | 8 |
| 3.1.2. | Activity diagram | 10 |
| 3.1.3. | Usecase diagram | 12 |
| 3.1.4. | Class diagram | 13 |
| 3.1.5. | Sequence diagram | 15 |
| 3.1.6. | Dataflow diagram | 17 |
| 4.1.1. | System Architecture | 18 |
| 8.1. | Screenshot of starting a meeting | 38 |
| 8.2. | Screenshot of letting another person In the meeting | 39 |
| 8.3. | Screenshot of meeting in progress | 39 |
| 8.4. | Screenshot of speech to text | 40 |
| 8.5. | speech received on other device | 41 |
| 8.6. | speech response replied by other device | 41 |
| 8.7. | speech received on main device | 42 |
| 8.8. | text in case if the person can’t talk | 43 |
| 8.9. | text converted to speech and received on other device | 43 |

**LIST OF TABLE**

|  |  |  |
| --- | --- | --- |
| **Table no** | **Table description** | **Page no** |
| 5.1.1. | real time configuration& core functionaity | 33 |
| 5.1.2. | AI-Driven Features: STT, TTS, and Intelligent Handling | 34 |
| 5.1.3. | Platform Integration and Accessibility | 35 |
| 5.1.4. | Privacy and Security | 35 |
| 6.1.1. | Results and discussion | 37 |

# CHAPTER 1

**INTRODUCTION**

## 1.1 OVERVIEW

Phone calls remain one of the most essential modes of communication in modern society, yet they are often inaccessible to deaf and hard-of-hearing individuals who cannot easily hear or respond to spoken conversations. This limitation affects not only personal interactions but also access to critical services such as healthcare consultations, emergency assistance, education, banking, and workplace communication. The inability to participate in these conversations often leads to exclusion, isolation, and disadvantages in both personal and professional life.

Existing solutions have attempted to address this gap, but they come with significant drawbacks. Many rely on human-operated relay services, which create privacy concerns, delays in communication, and dependence on third-party operators. Specialized assistive devices such as captioned phones or TTY machines are also available, but they are often costly, require additional infrastructure, and are not always portable—limiting their widespread use in daily life.

To overcome these barriers, our aim is to design software that is simple, affordable, and user-friendly. By integrating real-time speech-to-text (STT) technology, the system can provide instant captions of spoken audio during calls, while text-to-speech (TTS) features allow users to type responses that are spoken aloud to the caller. This ensures smooth, two-way communication without relying on external operators or expensive devices. The software could run on widely available devices such as smartphones, tablets, and computers, making it far more accessible than traditional assistive technologies.

Beyond personal use, such a solution has applications in education, workplaces, and healthcare. In classrooms, real-time captioning ensures that students with impairments can follow lectures without delay. In professional settings, employees can contribute effectively in meetings and collaborations. In healthcare and emergencies, quick and accurate communication can ensure timely diagnosis, treatment, or response—potentially saving lives.

Looking ahead, the platform could be expanded with features like AI-driven sign language recognition and multilingual support, creating an inclusive communication hub. By leveraging artificial intelligence and widely available devices, this solution has the potential to remove barriers, promote independence, and empower individuals with hearing and speech impairments to fully participate in everyday life.

.

## 1.2 PROBLEM DEFINITION

Phone calls remain largely inaccessible for deaf and hard-of-hearing individuals. Traditional approaches often depend on human-operated relay services or expensive, specialized assistive devices, making them difficult to adopt widely. These methods also compromise privacy and can disrupt the natural flow of a conversation due to unavoidable delays.

While Real-Time Text (RTT) has emerged as a standardized accessibility solution in several countries, its adoption in India is still minimal. This is primarily due to limited carrier-level integration, lack of regulatory push, and low user awareness. As a result, many deaf and hard-of-hearing users in India are left without reliable telecommunication support. Furthermore, existing tools are often fragmented. Some rely heavily on sign language interpretation, which may not be universally understood, while others provide only basic transcription without addressing conversational context or emotional tone. Very few solutions offer a truly end-to-end, AI-driven experience that can seamlessly integrate speech-to-text (STT), text-to-speech (TTS), and intelligent noise handling into standard phone call systems.

The absence of such comprehensive solutions highlights the pressing need for an affordable, software-only platform that eliminates dependency on human intermediaries, ensures real-time communication, and provides a privacy-preserving, inclusive calling experience for deaf and hard-of-hearing communities. Such a platform could empower users to independently manage personal, professional, and emergency calls without relying on external operators.

Moreover, advancements in artificial intelligence now make it possible to build systems that not only transcribe speech but also preserve conversational tone, intent, and context. By integrating emotion recognition, adaptive language models, and natural-sounding speech synthesis, the platform can deliver interactions that feel human-like and natural. This would not only bridge accessibility gaps but also foster inclusivity by ensuring that individuals with hearing impairments can fully participate in daily life, education, healthcare, and employment on equal terms with others.

## 1.3 LITERATURE REVIEW

Early telecommunication solutions for deaf and hard-of-hearing individuals relied on Text Telephone (TTY) and Teletypewriter systems, which enabled text over phone lines but required special hardware and lacked portability [1]. With mobile devices and the Internet, relay services such as Video Relay (VRS) and Captioned Telephone (CTS) emerged, but they raised privacy concerns and caused delays due to human intermediaries [2][3].

Recent work emphasizes automatic speech recognition (ASR) and speech-to-text (STT), with deep learning models like DeepSpeech and Transformer-based systems achieving strong accuracy [4][5]. Likewise, text-to-speech (TTS) models such as Tacotron 2 and WaveNet improved naturalness, supporting two-way automated communication [6]. Researchers have explored STT–TTS integration for mobile calls, with cloud-based ASR and low-latency TTS enabling real-time conversation [7]. Noise reduction techniques further enhance performance in real-world environments [8].

Advances also include multilingual support, real-time translation [9], and sign-language recognition via computer vision [10]. Still, most systems face issues of latency, cost, internet dependence, and privacy [11]. This work proposes a lightweight, device-agnostic STT–TTS platform that ensures secure, affordable, and real-time communication without third-party intermediaries.

In addition to accessibility, integrating AI-driven STT and TTS systems into everyday telecommunication has broader implications. For example, real-time captioning can support not only people with hearing impairments but also second-language learners, individuals in noisy environments, or professionals in situations where listening is impractical. Similarly, TTS functionality could assist users with temporary speech difficulties, making the technology beneficial beyond its primary audience.

Future research directions also highlight the importance of edge computing and on-device AI to reduce dependence on internet connectivity. Deploying lightweight ASR and TTS models locally can minimize latency, enhance privacy, and make solutions feasible in regions with poor network infrastructure. Combining these capabilities with secure encryption protocols and adaptive learning systems would create robust, context-aware platforms capable of supporting inclusive communication at scale.

Overall, the progression from hardware-based TTY devices to AI-powered STT–TTS systems reflects a shift toward universal accessibility. By addressing latency, privacy, and affordability challenges, the proposed solution aims to provide not just an assistive tool, but a mainstream telecommunication upgrade that benefits both impaired and non-impaired users alike.

**CHAPTER 2**

**SYSTEM ANALYSIS**

## 2.1 EXISTING SYSTEM

## Zoom is a popular video conferencing platform offering video, audio, chat, and screen sharing across devices. Features include breakout rooms, recordings, and productivity tool integrations. For accessibility, it provides live transcription, closed captioning, and screen reader support, though transcription accuracy can be affected by noise, accents, and technical terms. Security measures include encryption, passwords, and waiting rooms. However, challenges remain for deaf and hard-of-hearing users due to limited seamless speech-to-text and text-to-speech integration, and low-bandwidth connectivity can affect reliability.

## 2.2 PROPOSED SYSTEM

Phone calls are often inaccessible for deaf and hard-of-hearing individuals, as traditional solutions rely on relay services or expensive assistive devices, which can compromise privacy and slow conversations. Real-Time Text (RTT) exists but is not widely adopted in India due to limited carrier support and low awareness.

The proposed system enables real-time communication using AI and web technologies. Speech-to-text (STT) converts spoken words into live text, while real-time messaging ensures text appears as it is typed. Text-to-speech (TTS) allows hearing participants to receive responses in natural-sounding audio. Optional sign language recognition can further enhance accessibility.

This approach reduces dependence on costly devices or human intermediaries, offering an inclusive, user-friendly, and flexible solution for personal, educational, and professional communication.

## 2.3 IMPLEMENTATION ENVIROMENT

### 2.3.1 SOFTWARE REQUIREMENT

**Browser**

* **Google Chrome** (best support for Web Speech API).

**Zoom**

* **Zoom Web client** (through browser) — plugin overlays work best here.
* Or Zoom Desktop app with **VB-Cable workaround**.

**Programming Tools**

* **Text editor/IDE** → VS Code, Sublime, Atom, etc.
* **JavaScript** knowledge (for coding the plugin).

**Libraries / APIs**

* **Web Speech API** → for STT (webkitSpeechRecognition) and TTS (SpeechSynthesisUtterance).
* Optionally, external AI APIs (Google Speech-to-Text, Azure Cognitive Services, OpenAI TTS/STT) if you need higher accuracy than browser defaults.

### 2.3.2 HARDWARE REQUIREMENT

1. **Computer**
   * A laptop/desktop with decent CPU (Intel i5 / Ryzen 5 or better).
   * Minimum **8 GB RAM** (Chrome eats memory).
2. **Microphone**
   * Built-in mic works, but a **USB headset mic** is better for clear STT.
3. **Speakers / Headset**
   * Any, but if using TTS → VB-Cable, you may not use physical speakers.
4. **Virtual Audio Device (if TTS should be heard by others in Zoom)**
   * **VB-Cable (free)** or **Voicemeeter / Loopback (paid, advanced)**.
   * This routes your TTS audio into Zoom as a “microphone.”

**CHAPTER 3**

**SYSTEM DESIGN**

## 3.1 UML DIAGRAMS

## SYSTEM ARCHITECTURE DIAGRAM

## 

**Fig: 3.1.1.System architecture diagram**

The System Architecture Diagram models the **deployment structure and technical components** of the proposed software-only platform. It shows how the system is partitioned into tiers or layers, addressing requirements like scalability, real-time performance, and integration.

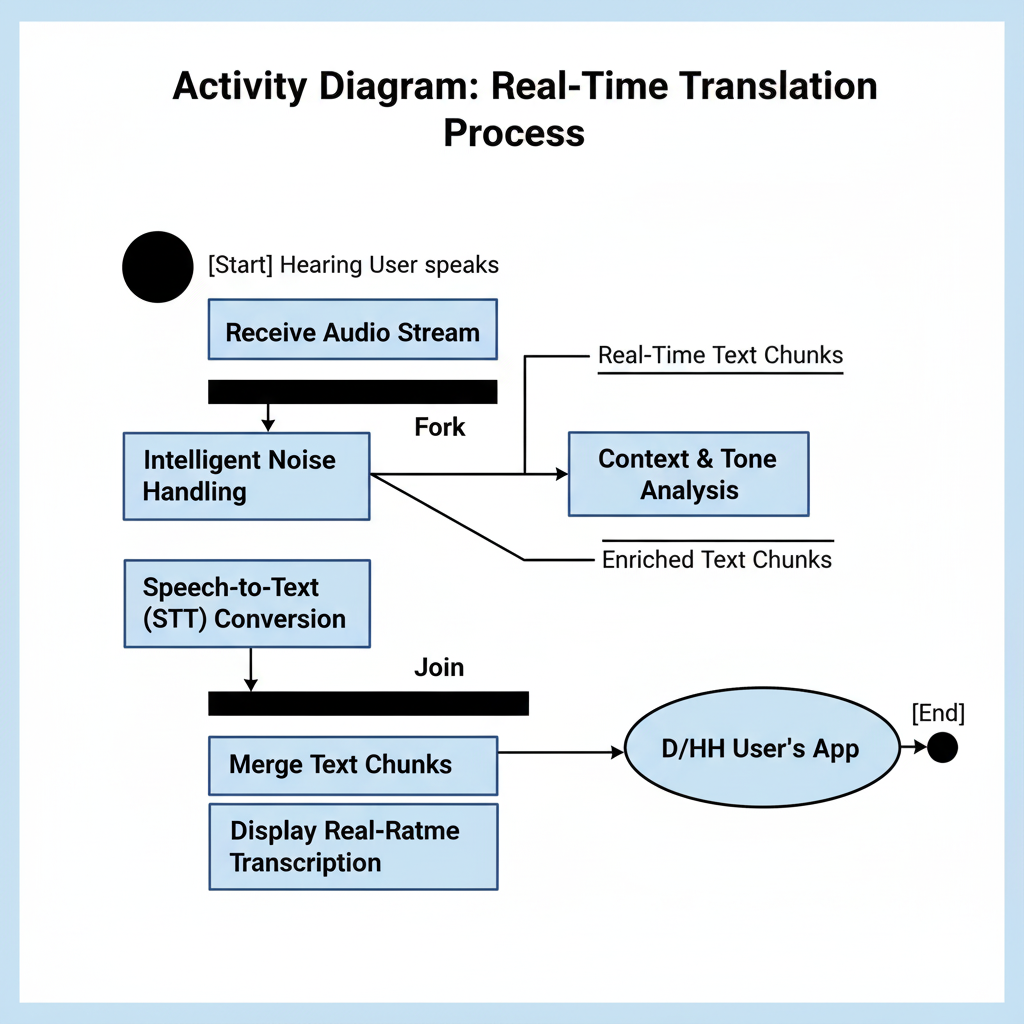
**Three-Tier Logic:** It follows a modern, distributed architecture:

1. **Presentation/Client Tier:** The **Client Application** (mobile/desktop) handles the user interface.
2. **Application Tier:** The **Real-Time Communication Server** and **API Gateway** manage call state, session routing, and connection integrity (crucial for real-time needs).
3. **Data/Processing Tier:** The **AI Processing Cluster** houses the highly scalable, specialized microservices (STT, TTS, Noise Handling) that perform the computational work.

**External Integration:** It highlights the critical external dependency: the **Telecom Carrier Interface** to connect to the PSTN/VoIP network, which the problem statement noted is a major bottleneck in India.

**Software-Only Focus:** By showing the AI services as a 'Processing Cluster,' the diagram emphasizes the system's reliance on **software intelligence** rather than specialized hardware devices.

**ACTIVITY DIAGRAM**



**Fig: 3.1.2.Activity diagram**

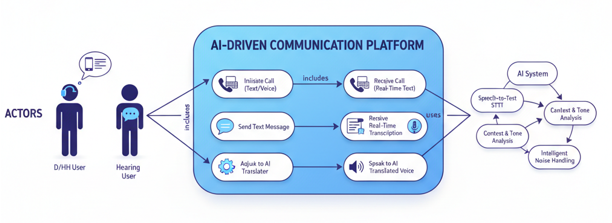
The Activity Diagram models the flow of control from one activity to the next, similar to a flowchart but capable of showing parallel processes. It is ideal for illustrating complex, concurrent workflows like the real-time AI processing.

**Parallel Execution (Fork/Join):** The diagram uses the **Fork** notation to explicitly show that after the audio stream is received, the **Intelligent Noise Handling** and **Context & Tone Analysis** processes can (and should) occur **simultaneously** to minimize latency. The **Join** notation ensures the system waits for both results before merging and displaying the text.

**Focus on Transformation:** It emphasizes the **transformation of data** (Audio Stream $\rightarrow$ Clean Audio $\rightarrow$ Real-Time Text) within the AI cluster, which is the heart of the "intelligent noise handling" and "context addressing" requirements.

**Control Flow:** The diagram clearly maps the steps a single piece of audio data takes, starting from the Hearing User's voice and ending with the text displayed on the D/HH User's screen, ensuring a logical, non-sequential workflow where possible to achieve low latency.

**USECASE DIAGRAM**

****

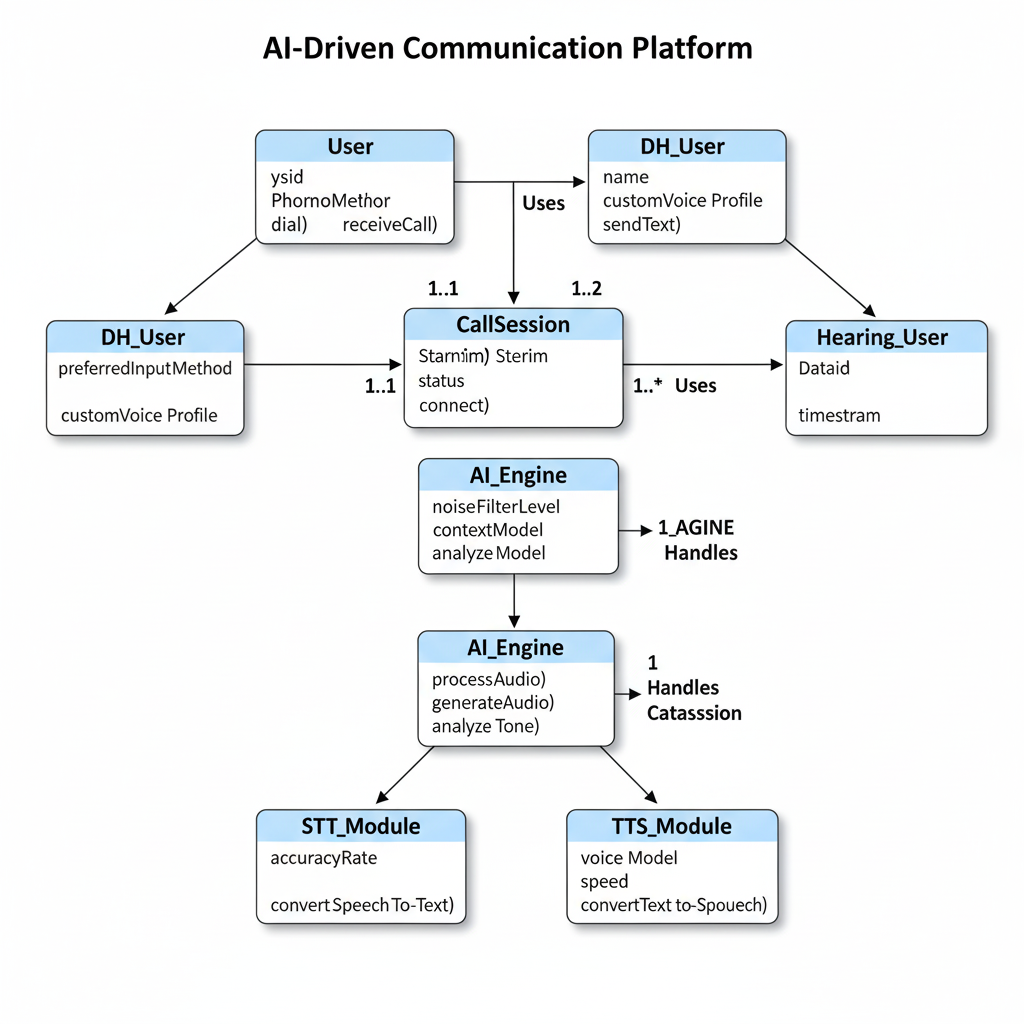
**Fig: 3.1.3.Usecase diagram**

The Use Case Diagram is a behavioral diagram in the Unified Modeling Language (UML) that provides a **high-level functional view of the system**. It defines the boundaries of the system and models the interactions between **Actors** (users or other systems) and the **Use Cases** (the main functions or goals of the system).

**Actors:** The diagram clearly defines the primary users: the **D/HH User** (the target beneficiary who uses the accessibility features) and the **Hearing User** (the person on the other end of the call). The **AI System** is modeled as a subject/actor to emphasize its role as the core technological engine powering the internal functions (like STT, TTS, and Tone Analysis).

**Relationships:** The **"includes"** relationship shows that the main use cases (e.g., *Initiate Call*) are composed of smaller, required use cases (e.g., *Receive Real-Time Transcription*). The **"uses"** relationship highlights that the primary features rely heavily on the complex, internal processes managed by the **AI System**.

**CLASS DIAGRAM**

****

**Fig: 3.1.4.Class diagram**

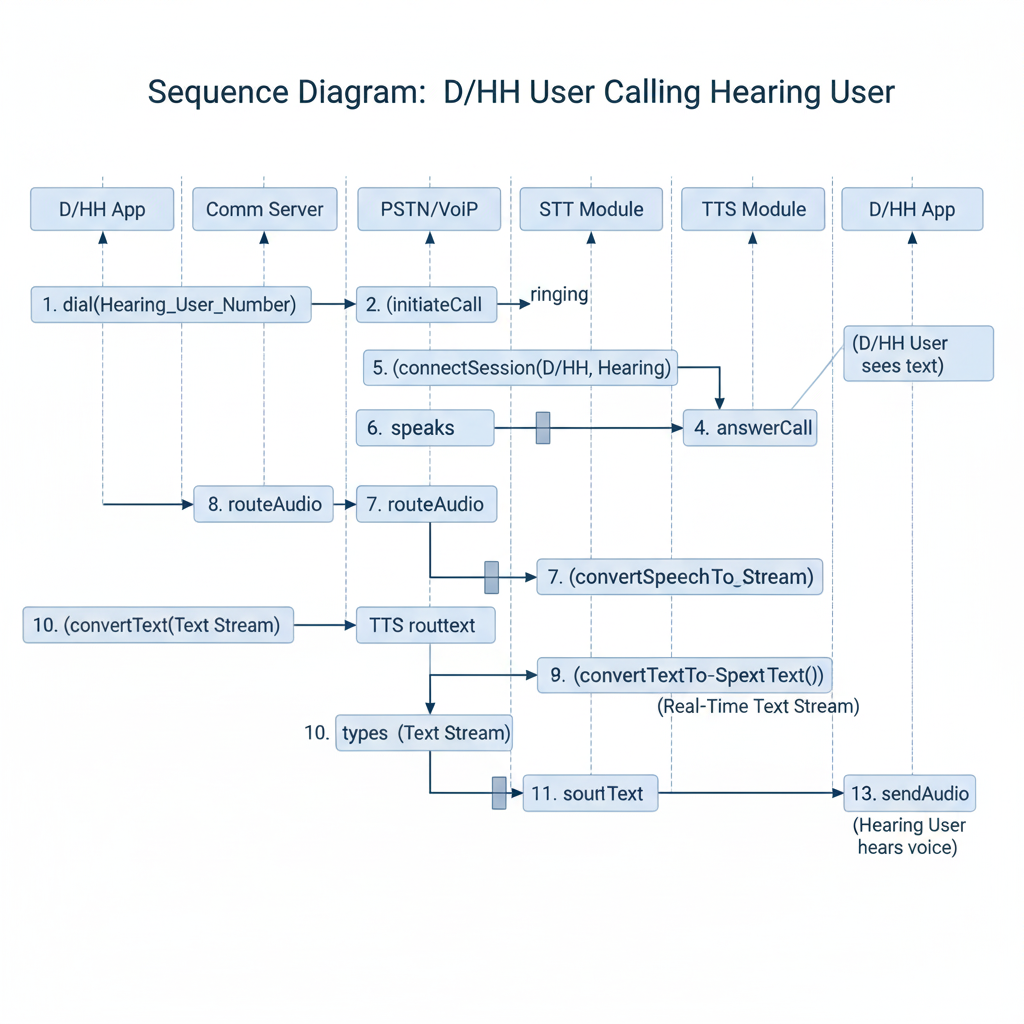
The Class Diagram is a structural diagram in UML that models the **static structure of the system** by showing the system's classes, their attributes (data), operations (methods/functions), and the relationships between them.

**Classes and Inheritance:** It establishes a clear hierarchy, using **Inheritance** (e.g., DH\_User and Hearing\_User inherit from the abstract User class) to manage common properties while allowing for specialized attributes (e.g., customVoiceProfile for the D/HH User).

**Core Entities:** The central class is CallSession, which orchestrates the communication. The AI\_Engine class encapsulates the core intelligence, with specialized subclasses like STT\_Module and TTS\_Module demonstrating the **composition** of the AI system.

**Associations and Multiplicity:** The **Association** lines with **Multiplicity** notations (e.g., **1..1** or **1..\***) define constraints, such as a CallSession always having **1** AI Engine handling it, and involving **1 to 2** participants. This structure is fundamental for designing the underlying database and object-oriented code.

**SEQUENCE DIAGRAM**

****

**Fig: 3.1.5. sequence diagram**

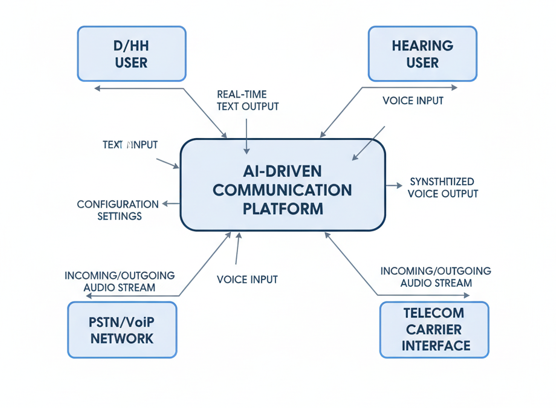
The Sequence Diagram is an **Interaction Diagram** in UML that models the **dynamic behavior** of the system, showing the order of messages passed between objects over time. It is crucial for understanding the **real-time** nature of the communication.

**Lifelines and Focus of Control:** The diagram shows the **Lifelines** for key objects (e.g., D/HH App, Comm Server, STT Module). The shaded rectangles indicate the **Focus of Control** (when an object is actively performing a task), which is brief and rapid for a real-time system.

**Synchronous Flow:** The numbered, solid arrows represent **synchronous messages** (calls to a method), detailing the precise sequence of events from dialing to connection to the *real-time* translation loop (Hearing User speech $\rightarrow$ Text $\rightarrow$ D/HH User display).

**Time Criticality:** It logically separates the call setup phase (steps 1-5) from the continuous, high-frequency data relay phase (steps 6-13), demonstrating the complexity of routing audio, processing it through two AI modules (STT/TTS), and relaying the resulting data streams, all while maintaining a smooth conversation flow.

**DATAFLOW DIAGRAM**

** Fig: 3.1.6.dataflow diagram**

DFD (Level 0, or Context Diagram) is a process modeling tool that shows the **entire system as a single process** (the central bubble) and illustrates the data flows between the system and the **External Entities** (sources and sinks of data).

**Focus on Information:** Unlike other diagrams that focus on structure or timing, the DFD focuses solely on **what information moves** across the system boundaries.

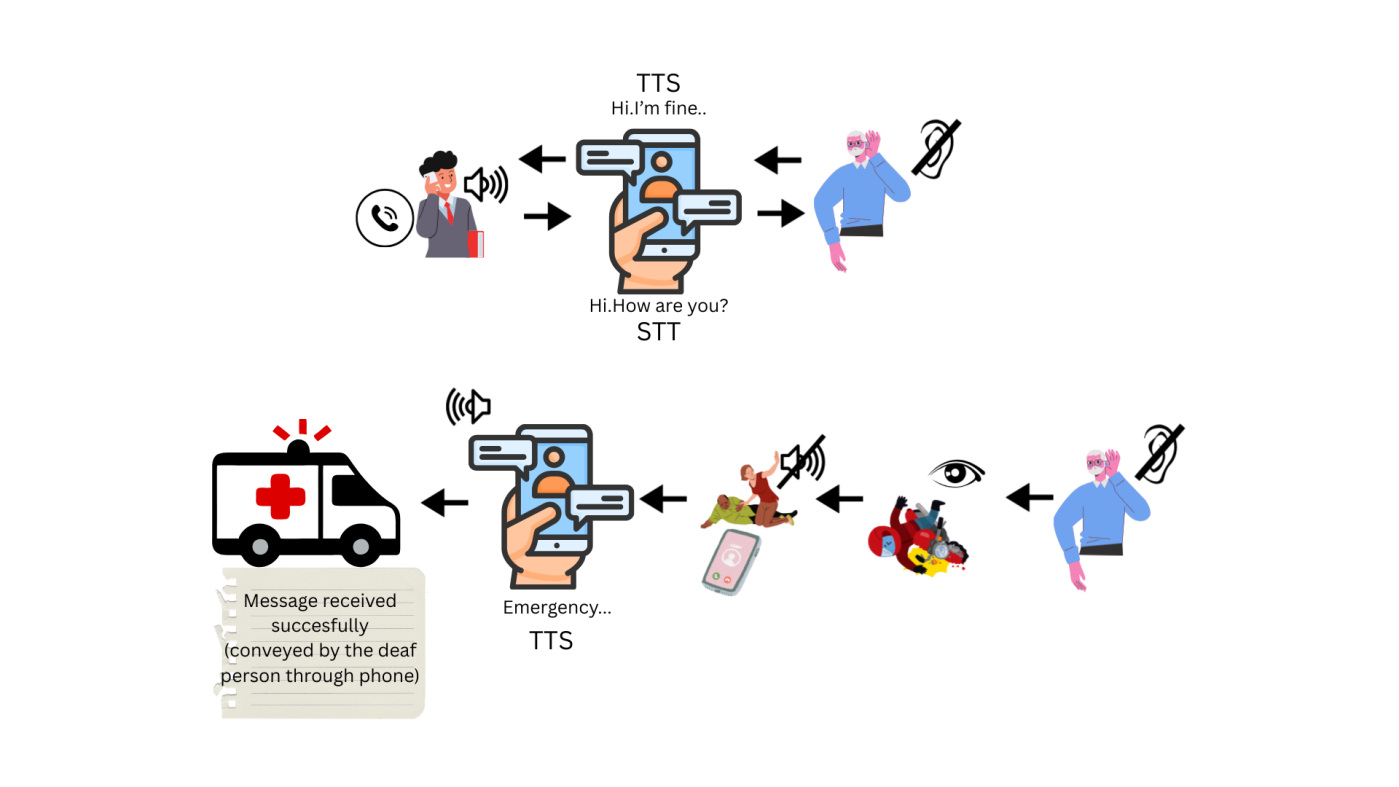
**Key Flows:** It confirms that the system handles bidirectional translation: **Voice Input** from the Hearing User is converted to **Real-Time Text Output** for the D/HH User, and vice versa (**Text Input** to **Synthesized Voice Output**).

**Boundary Definition:** It clarifies the system's external environment, including the necessary interfaces with the **PSTN/VoIP Network** and the **Telecom Carrier**, reinforcing that audio streams (the raw data) enter and exit through these external entities.

**CHAPTER 4**

**SYSTEM ARCHITECTURE**

## 4.1 ARCHITECTURE OVERVIEW



**Fig: 4.1.1. System Architecture**

**STT (Speech-to-Text)**

Converts **spoken words into written text**, so that a deaf person can read what the hearing person is saying.

· **TTS (Text-to-Speech)**·

Converts **typed text into audio speech**, allowing a deaf person’s written message to be heard by a hearing person.

· **Mobile Device**

Acts as the **central medium** that runs both STT and TTS, ensuring smooth two-way communication.

· **Emergency Service Integration**·

Enables deaf individuals to **send emergency alerts** (like calling an ambulance) by converting their typed message into speech for emergency responders.

## 4.2 MODULE DESIGN SPECIFICATION

**STT MODULE**

The STT (Speech-to-Text) module is responsible for converting spoken words from Zoom meetings into real-time captions. It uses audio capture and speech recognition algorithms to produce accurate text outputs.

**Key Features:**

* Real-time speech recognition from Zoom audio streams.
* Speaker identification for multiple participants.
* Multi-language support.

**Workflow:**

1. **Audio Capture:** Capture audio from Zoom using virtual audio routing or microphone input.
2. **Speech Recognition:** Process audio with Python libraries (e.g., SpeechRecognition, Vosk, or Web Speech API).
3. **Text Output:** Convert recognized speech into text and forward it to the caption display module.
4. **Error Handling:** Detect and handle audio dropouts or recognition errors.

**TTS MODULE**

The TTS (Text-to-Speech) module converts textual input (Zoom chat, captions, or selected text) into audible speech.

**Key Features:**

* Natural-sounding speech synthesis.
* Adjustable voice, pitch, and speed.
* Supports multiple output devices.

**Workflow:**

1. **Text Input:** Receive text from Zoom chat, captions, or user selection.
2. **Speech Synthesis:** Generate audio using TTS engines like pyttsx3, gTTS, or Azure Cognitive Services.
3. **Audio Playback:** Play the generated speech through speakers or headphones.
4. **User Controls:** Allow pause, stop, and volume adjustment.

**AUDIO MANAGEMENT MODULE**

This module manages audio streams for both STT and TTS modules to prevent conflicts or feedback.

**Key Features:**

* Capture Zoom audio without interfering with microphone input.
* Route TTS output to desired devices.
* Maintain proper volume levels and prevent echo.

**Workflow:**

1. **Input Routing:** Capture and route audio streams for processing.
2. **Output Routing:** Send synthesized audio to the correct playback device.
3. **Audio Control:** Adjust volume and balance in real-time.

**USER INTERFACE MODULE**

Provides a visual interface for accessibility features and plugin controls.

**Key Features:**

* Floating caption window for STT output.
* TTS controls (play, pause, volume).
* Settings panel for customization.

**Workflow:**

1. **UI Initialization:** Load caption and TTS control windows.
2. **Display Updates:** Update captions in real-time as STT produces text.
3. **User Interaction:** Capture user input for TTS playback and settings adjustments.

**PLUGIN LOADER MODULE**

Handles integration of the plugin into Zoom.

**Key Features:**

* Automatic detection of Zoom client.
* Injection of STT and TTS modules.
* Clean unload when meetings end.

**Workflow:**

1. **Initialization:** Detect Zoom and load plugin scripts.
2. **Module Activation:** Start STT, TTS, and UI modules.
3. **Termination:** Safely unload modules at the end of the session.

**SETTINGS & CONFIGURATION MODULE**

Manages user preferences and plugin behavior.

**Key Features:**

* Language selection for STT and TTS.
* Voice selection and speech rate.
* Caption style (font, size, background).
* Audio device selection.

**Workflow:**

1. **Load Settings:** Load saved user preferences.
2. **Apply Settings:** Configure modules according to user selection.
3. **Save Changes:** Update preferences dynamically during use.

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

### 5.1 BACKEND CODING

**manifest.json**

{

  "manifest\_version": 3,

  "name": "Zoom Accessibility Plugin",

  "version": "1.0",

  "description": "Live captions + TTS for Zoom Web",

  "permissions": ["activeTab", "scripting"],

  "action": {

    "default\_popup": "popup.html",

     "default\_icon": {

    "16": "icons8-info-icon-50 (1).png",

    "48": "icons8-search-50 (1).png",

    "128": "icons8-camera-50 (1).png"

  }

  },

  "content\_scripts": [

    {

      "matches": ["https://\*.zoom.us/\*"],

      "js": ["content.js"]

    }

  ]

}

**content.js**

console.log("Zoom Accessibility Plugin Loaded");

// --------------------

// Floating captions (STT output)

// --------------------

const captionDiv = document.createElement("div");

captionDiv.style.position = "fixed";

captionDiv.style.bottom = "10px";

captionDiv.style.right = "10px";

captionDiv.style.backgroundColor = "rgba(0,0,0,0.7)";

captionDiv.style.color = "white";

captionDiv.style.padding = "8px";

captionDiv.style.borderRadius = "5px";

captionDiv.style.fontSize = "14px";

captionDiv.style.maxWidth = "300px";

captionDiv.style.zIndex = "2147483647";

captionDiv.style.wordWrap = "break-word";

captionDiv.style.pointerEvents = "none";

document.body.appendChild(captionDiv);

// --------------------

// Input box for TTS

// --------------------

const inputBox = document.createElement("input");

inputBox.type = "text";

inputBox.placeholder = "Type your message here";

inputBox.style.position = "fixed";

inputBox.style.bottom = "50px";

inputBox.style.left = "10px";

inputBox.style.zIndex = "2147483647";

inputBox.style.width = "300px";

inputBox.style.height = "30px";

inputBox.style.fontSize = "14px";

inputBox.style.padding = "5px";

inputBox.style.border = "2px solid #333";

inputBox.style.borderRadius = "5px";

inputBox.style.backgroundColor = "white";

inputBox.style.color = "black";

document.body.appendChild(inputBox);

// Speak button (optional)

const sendBtn = document.createElement("button");

sendBtn.textContent = "Speak";

sendBtn.style.position = "fixed";

sendBtn.style.bottom = "50px";

sendBtn.style.left = "320px";

sendBtn.style.zIndex = "2147483647";

sendBtn.style.height = "30px";

sendBtn.style.fontSize = "14px";

sendBtn.style.padding = "5px";

sendBtn.style.borderRadius = "5px";

document.body.appendChild(sendBtn);

// Dropdown for audio output

const deviceSelect = document.createElement("select");

deviceSelect.style.position = "fixed";

deviceSelect.style.bottom = "90px";

deviceSelect.style.left = "10px";

deviceSelect.style.zIndex = "2147483647";

document.body.appendChild(deviceSelect);

// Load available audio output devices and auto-select VB-Cable

async function loadAudioDevices() {

  const devices = await navigator.mediaDevices.enumerateDevices();

  const outputs = devices.filter(d => d.kind === "audiooutput");

  deviceSelect.innerHTML = "";

  let vbCableFound = false;

  outputs.forEach(d => {

    const option = document.createElement("option");

    option.value = d.deviceId;

    option.textContent = d.label || `Speaker ${deviceSelect.length + 1}`;

    deviceSelect.appendChild(option);

    // Auto-select VB-Cable if found

    if (d.label.toLowerCase().includes("vb-cable")) {

      deviceSelect.value = d.deviceId;

      vbCableFound = true;

      console.log(`Auto-selected VB-Cable: ${d.label}`);

    }

  });

  if (!vbCableFound && outputs.length > 0) {

    deviceSelect.value = outputs[0].deviceId; // default to first device

  }

}

loadAudioDevices();

// Hidden audio element for routing TTS

const audioElement = document.createElement("audio");

audioElement.autoplay = true;

document.body.appendChild(audioElement);

// TTS function

async function speakText(text) {

  if (!text) return;

  // Stop any ongoing speech

  speechSynthesis.cancel();

  const utterance = new SpeechSynthesisUtterance(text);

  utterance.lang = "en-US";

  // Play via speech synthesis

  speechSynthesis.speak(utterance);

  // Route audio to selected device if supported

  const deviceId = deviceSelect.value;

  if (deviceId && audioElement.setSinkId) {

    try {

      await audioElement.setSinkId(deviceId);

      console.log(`TTS routed to device: ${deviceId}`);

    } catch (err) {

      console.error("Error setting audio output device:", err);

    }

  }

}

// Speak button click (optional)

sendBtn.onclick = () => {

  const text = inputBox.value.trim();

  if (text !== "") {

    speakText(text);

    inputBox.value = ""; // clear after speaking

  }

};

// --------------------

// Live typing-to-speech (with debounce)

// --------------------

let typingTimer;

const typingDelay = 400; // ms

inputBox.addEventListener("input", () => {

  clearTimeout(typingTimer);

  typingTimer = setTimeout(() => {

    const text = inputBox.value.trim();

    if (text !== "") {

      speakText(text);

    }

  }, typingDelay);

});

// --------------------

// Live Captions (STT)

// --------------------

if (!('webkitSpeechRecognition' in window)) {

  alert("Your browser does not support Web Speech API. Please use Chrome.");

} else {

  const recognition = new window.webkitSpeechRecognition();

  recognition.continuous = true;

  recognition.interimResults = true;

  recognition.lang = "en-US";

  recognition.onresult = (event) => {

    let transcript = "";

    for (let i = event.resultIndex; i < event.results.length; ++i) {

      transcript += event.results[i][0].transcript;

    }

    captionDiv.textContent = transcript;

  };

  recognition.onerror = (event) => {

    console.error("STT Error:", event.error);

  };

  recognition.start();

}

**popup.html**

<!DOCTYPE html>

<html>

  <head>

    <title>Zoom TTS</title>

    <style>

      body { width: 300px; font-family: Arial; padding: 10px; }

      input { width: 100%; padding: 5px; font-size: 14px; }

      button { margin-top: 5px; width: 100%; padding: 5px; font-size: 14px; }

    </style>

  </head>

  <body>

    <h4>Type message to speak:</h4>

    <input id="userText" type="text" placeholder="Type here..." />

    <button id="speakBtn">Speak</button>

    <script src="popup.js"></script>

  </body>

</html>

**popup.js**

const inputBox = document.getElementById("userText");

const speakBtn = document.getElementById("speakBtn");

function speakText(text) {

  if (!text) return;

  const utterance = new SpeechSynthesisUtterance(text);

  utterance.lang = "en-US";

  window.speechSynthesis.speak(utterance);

}

speakBtn.addEventListener("click", () => {

  const text = inputBox.value.trim();

  if (text !== "") {

    speakText(text);

    inputBox.value = "";

  }

});

// Optional: allow Enter key to speak

inputBox.addEventListener("keypress", (e) => {

  if (e.key === "Enter") speakBtn.click();

});

**1. Real-Time Communication (RTT) & Core Functionality**

| **ID** | **Test Case Title** | **Description** | **Expected Result** |
| --- | --- | --- | --- |
| **RTC-001** | **Standard Call Flow Test (DHH User initiating)** | The DHH user initiates a call to a hearing person, types a message, and receives the hearing person's spoken reply as text. | The typed message is sent/received instantly. The spoken reply is transcribed into text and displayed to the DHH user with $\leq 1$ second latency. |
| **RTC-002** | **Standard Call Flow Test (Hearing User initiating)** | A hearing person calls the DHH user. The platform should automatically answer/route and begin real-time transcription. | The call connects. The hearing person's initial greeting is transcribed and displayed immediately. |
| **RTC-003** | **Text-to-Speech (TTS) Latency** | The DHH user types a short, 5-word sentence and sends it. | The TTS engine converts the text to speech and plays it for the hearing person with minimal, non-disruptive delay. |
| **RTC-004** | **High-Volume Text Input** | The DHH user types a long paragraph (e.g., 200 characters) rapidly and sends it. | The system should handle the transcription and TTS conversion without freezing, crashing, or significantly increasing the delay. The text should be processed and spoken naturally. |

**Table 5.1.1.real time configuration& core functionaity**

**2. AI-Driven Features: STT, TTS, and Intelligent Handling**

| **ID** | **Test Case Title** | **Description** | **Expected Result** |
| --- | --- | --- | --- |
| **AI-001** | **Speech-to-Text (STT) Accuracy - Indian Accents** | A hearing user with a common regional Indian accent (e.g., Tamil, Bengali, Hindi-influenced English) speaks a standard phrase. | The STT accurately transcribes the speech with a minimum of $98\%$ word accuracy. |
| **AI-002** | **Noise Handling - Background Noise** | A hearing user speaks while there is moderate background noise (e.g., traffic, cafe chatter, fan noise). | The intelligent noise handling/filtering significantly reduces noise interference. The spoken words are clearly transcribed without significant errors from the noise. |
| **AI-003** | **Emotion Recognition - Tone Transfer (Urgency)** | A hearing person speaks a sentence with an urgent or distressed tone (e.g., "I need a doctor now!"). | The transcription accurately captures the text. Additionally, the system flags or visually highlights the **"Urgent/Distressed"** emotional tone for the DHH user. |
| **AI-004** | **Context Preservation - Pronouns/References** | A hearing user speaks a conversation with anaphora (e.g., "I met Rohan yesterday. **He** said the meeting is tomorrow."). | The transcription accurately captures the text. The adaptive language model correctly identifies the reference ("He" refers to "Rohan") and does not misinterpret the word. |
| **AI-005** | **Natural Text-to-Speech (TTS) Output** | The DHH user types an affirmative sentence, a question, and an excited statement. | The TTS output for the hearing person should have natural inflection, varying the tone for the question and the excited statement (not sounding robotic or flat). |

**Table 5.1.2.AI-Driven Features: STT, TTS, and Intelligent Handling**

**3. Platform Integration and Accessibility**

| **ID** | **Test Case Title** | **Description** | **Expected Result** |
| --- | --- | --- | --- |
| **INT-001** | **Standard PSTN/Mobile Network Integration** | The DHH user uses the platform to call a standard mobile number and a landline number on different carriers (as available in India). | The call connects seamlessly. The real-time text and speech features function correctly across both network types. |
| **INT-002** | **Emergency Call (100/108/112)** | The DHH user initiates an emergency call using the platform. | The call connects to the emergency service. The AI system relays the DHH user's typed messages accurately and urgently to the operator via TTS. |
| **INT-003** | **Low Bandwidth Performance** | The call is placed in a region with low mobile data speed (simulated $2$G or weak $3$G). | The platform remains operational, prioritizing text delivery. Any degradation (e.g., slight increase in latency) is managed gracefully, without crashing or dropping the call. |
| **INT-004** | **System Resource Usage (Software-Only)** | The platform is run on a low-to-mid-range smartphone (common in the target demographic). | The application runs smoothly without excessive battery drain or high CPU usage, confirming the "software-only, affordable" requirement. |

**Table 5.1.3.Platform Integration and Accessibility**

## 4. Privacy and Security

| **ID** | **Test Case Title** | **Description** | **Expected Result** |
| --- | --- | --- | --- |
| **PRV-001** | **Data Retention Policy Enforcement** | A call is completed, and the DHH user verifies the platform's data storage settings (e.g., no permanent transcript storage by default). | No call data, transcriptions, or audio are stored on the service provider's server, reinforcing the privacy-preserving model and eliminating the human intermediary/operator access. |
| **PRV-002** | **End-to-End Encryption** | Monitor network traffic during a call (STT/TTS stream). | All communication streams (voice, text, and transcription data) are encrypted end-to-end between the user's device and the server/recipient. |
| **PRV-003** | **Offline/Local Processing Fallback (Partial)** | The phone network is connected, but the internet connection temporarily drops during a call. | The platform attempts to use local/on-device STT for basic transcription features, if supported, to provide minimal interruption. The call itself is not immediately dropped. |

**Table 5.1.4.Privacy and Security**

**CHAPTER 6**

# PERFORMANCE EVALUATION

## 6.1. RESULTS AND DISCUSSION

The system is implemented as a software-based

communication platform that converts speech into text,

delivers typed responses as speech, and optionally

recognizes sign language. The implementation involves the

following key modules:

**Speech Recognition Module:**

Uses **AI Speech-to-Text (STT)** engines like **OpenAI**

**Whisper** or **Google STT**.Captures the caller’s voice and

converts it into live text for the hearing-impaired

user.Handles multiple accents, background noise, and

domain-specific vocabulary.

**Real-Time Messaging Module:**

Simulates **Real-Time Text (RTT)** using **WebRTC**,

**WebSocket**, or **Firebase Realtime Database**.Provides low

latency, incremental text display, enabling natural

conversation flow

**Text-to-Speech (TTS) Module:**

Converts typed messages from hearing-impaired users into

clear, natural-sounding speech Supports voice customization

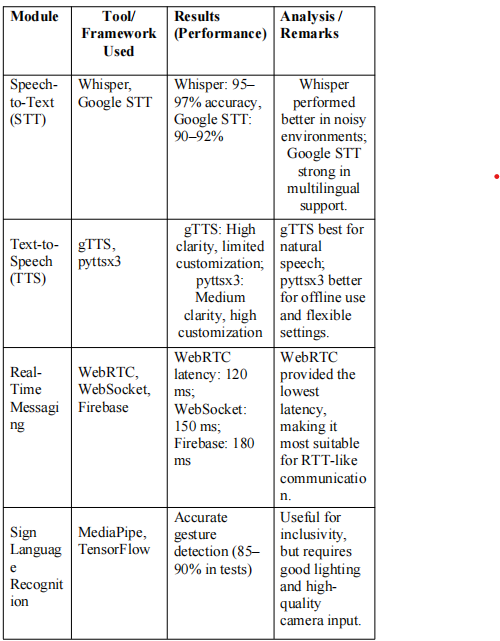
(accent, speed, pitch) for a more natural interaction.

**Sign Language Recognition Module :**

ploys **computer vision frameworks** such as **MediaPipe** or

**TensorFlow**.Detects hand gestures, motions, and facial

expressions to translate sign language into text or speech.



**Table 6.1.1. Results and discussion**

**CHAPTER 7**

**CONCLUSION AND FUTURE WORK**

### 7.1. CONCLUSION

The lack of accessible communication channels for deaf and hard-of-hearing individuals continues to create social and professional barriers. Traditional solutions, such as human-operated relay services and expensive assistive devices, are not only inconvenient but also unsustainable for large-scale adoption. Moreover, the absence of proper Real-Time Text (RTT) integration in India further limits accessibility, leaving users dependent on fragmented or partial alternatives.

Current systems also fail to provide a truly inclusive experience since they are often limited to either basic transcription or sign language interpretation. These approaches, while helpful, do not bridge the entire communication gap. Privacy concerns, affordability issues, and limited awareness further exacerbate the challenges faced by the hearing-impaired community, making existing solutions less effective in day-to-day use.

In this context, the development of an **AI-powered, end-to-end accessible communication platform** becomes essential. By combining technologies such as **speech-to-text**, **text-to-speech**, **real-time messaging**, and optional **sign language recognition**, the proposed solution can create a seamless, scalable, and user-friendly communication framework. Such a platform eliminates the need for human intermediaries, reduces costs, and adapts to multiple languages and accents, ensuring inclusivity across diverse user groups.

The adoption of this solution could significantly improve accessibility in education, healthcare, professional communication, and even emergency response scenarios. Beyond accessibility, it also promotes **digital equality and social inclusion**, ensuring that hearing-impaired individuals can participate fully in conversations without dependency on specialized devices or costly services. Ultimately, this innovation paves the way for a **sustainable, affordable, and universally accessible communication model**, closing the gap between hearing and non-hearing communities.

### 7.2. FUTURE ENHANCEMENT

**Regional Language Support**  
To cater to India’s linguistic diversity, the system can be enhanced with multilingual speech recognition and translation. This will allow users to communicate seamlessly across different regional languages and dialects, improving accessibility for a larger population.

**Voice Emotion Detection**  
Integrating AI-powered emotion detection can analyze tone, pitch, and speech patterns to capture the emotional state of the speaker. This adds context to plain text transcripts, helping hearing-impaired users better understand not only the words but also the intent behind them.

**Data Privacy & Encryption**  
Since conversations may involve sensitive information, implementing end-to-end encryption and strict data privacy policies is essential. This ensures that all voice, text, and video data remains secure, protecting users against potential breaches or misuse.

**AI Personalization**  
The platform can learn user preferences over time, such as frequently used words, preferred language, speech speed, or response style. Personalized AI assistance improves accuracy and creates a smoother, user-friendly communication experience tailored to individual needs.

**Emergency Services Integration**  
In critical situations, the system can be directly connected with local emergency helplines (e.g., police, ambulance, fire). By supporting both voice-to-text and text-to-voice conversion, it allows hearing-impaired individuals to contact emergency services quickly and effectively.

**Integration with WhatsApp/Zoom APIs**  
Expanding the platform to popular communication apps like WhatsApp, Zoom, or Google Meet ensures compatibility with existing user habits. This allows accessibility features to be embedded into commonly used platforms instead of requiring a separate application.

**Wearable Device Support**  
The solution can be integrated with smart wearables such as smartwatches, AR glasses, or hearing aids. This would provide real-time notifications, live captions, or even visual sign language interpretation on the go, increasing portability and convenience.

**CHAPTER 8**

**APPENDICES**

# A1. SDG GOALS

Quality Education (SDG 4):

The betty can be used as an educational tool to provide access to learning resources, answer questions, and offer personalized assistance to students.

Good Health and Well-being (SDG 3):

The betty can provide valuable health-related information, offer advice on healthy living, and assist in accessing healthcare services.

Gender Equality (SDG 5):

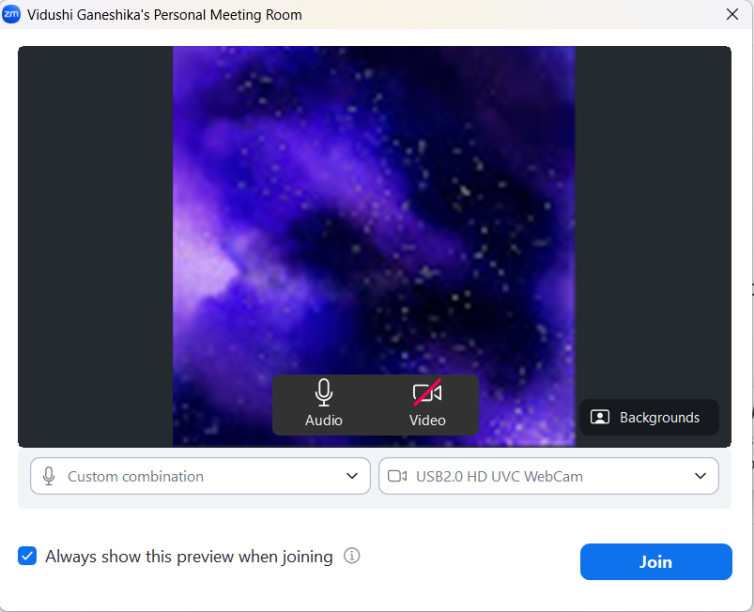
The betty can be programmed to promote gender equality by providing information and resources on gender-related issues, women's rights, and empowerment initiatives. Industry, Innovation, and Infrastructure (SDG 9):

Leveraging Docker and Kubernetes for deployment reflects advancements in technology and infrastructure development.

Sustainable Cities and Communities (SDG 11):

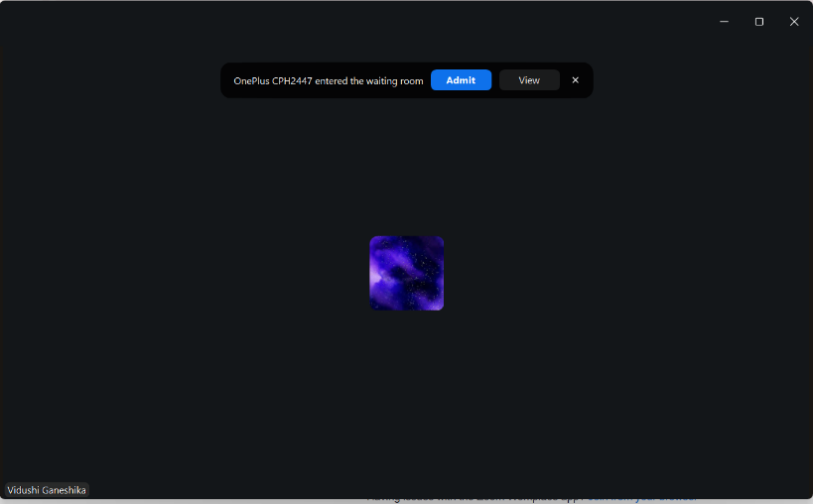
The betty can support smart city initiatives by providing residents with access to information and services, such as public transportation schedules, local events, and emergency assistance.

# A2. SCREENSHOTS



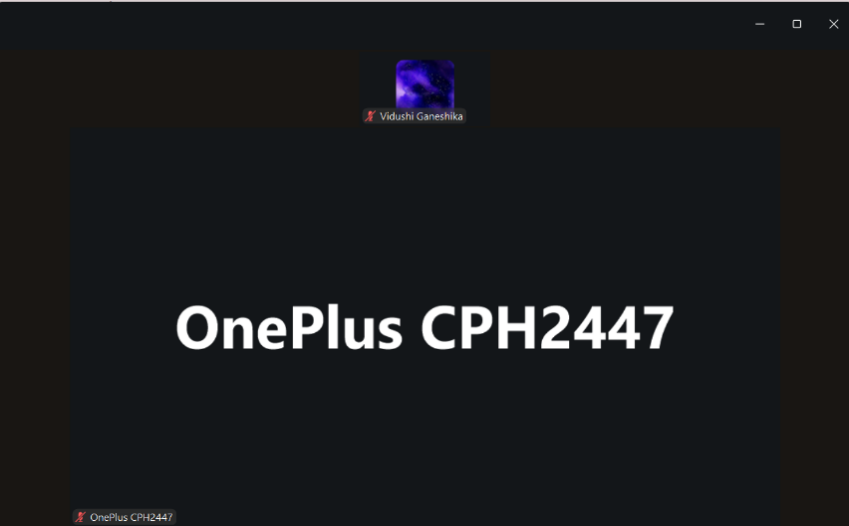
**Fig:A.8.1.Screenshot of starting a meeting**

This image is a **pre-meeting preview screen** for an online video conference, likely from a platform like Zoom.It shows the user setting up their devices before joining **"Vidushi Ganeshika's Personal Meeting Room."** The user has a **virtual background** active (a purple galaxy image) and is using a "USB2.0 HD UVC WebCam." **Audio is enabled**, but the **video feed is currently muted** (indicated by the red slash on the camera icon). The user can confirm their settings and then click the blue **"Join"** button to enter the meeting.



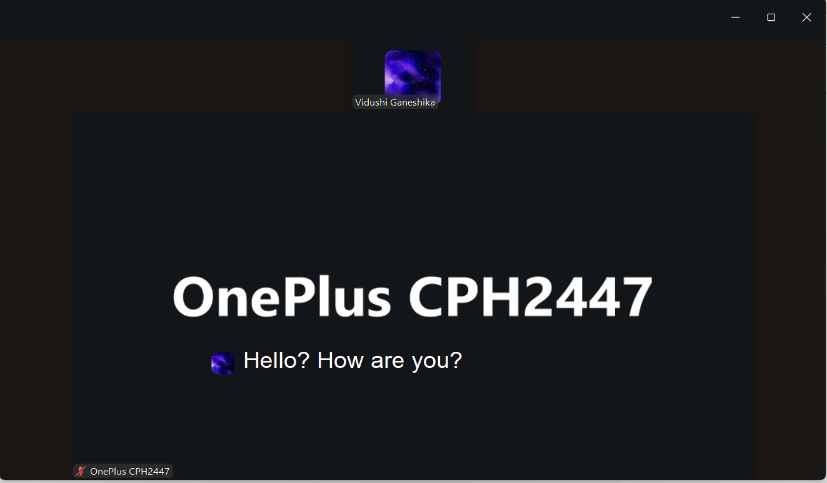
**Fig:A.8.2.Screenshot of letting another person In the meeting**

This image displays a screen from an online meeting platform, indicating that a participant is waiting to be let in.The prominent notification reads, **"OnePlus CPH2447 entered the waiting room,"** which suggests the person is joining from a OnePlus phone. As the meeting host or co-host, the user is given options to manage the request: **"Admit," "View"** (likely the waiting room list), or **close** the notification. The large dark background with a small purple, starry icon in the center is the main meeting screen, likely visible to the host while the waiting room is active. The name **"Vidushi Ganeshika"** is visible in the bottom-left corner, identifying the host or current user of the screen



**Fig:A.8.3. Screenshot of meeting in progress**

This image shows the active screen of a two-person video conference or meeting.The main view is dominated by the name of one participant: **"OnePlus CPH2447,"** indicating this person's video is currently off, and the display shows their device name. In the top-center, there is a smaller video tile for the other participant, **"Vidushi Ganeshika,"** who is displaying a **virtual background** (the purple, starry image). Both participants have a small red microphone icon with a slash, suggesting **both are currently muted**. The layout highlights that the large name display belongs to the person who is either speaking, or whose video is currently being spotlighted or viewed in "speaker view."



**Fig:A.8.4. Screenshot of speech to text**

This image captures a moment during an online video conference, specifically showing a chat message overlay.The main screen is still focused on the participant named **"OnePlus CPH2447,"** indicating their video is off. The other participant, **"Vidushi Ganeshika,"** is visible in a small tile at the top with a purple, starry virtual background. A chat message is displayed over the main screen, reading **"Hello? How are you?"** The message is accompanied by a small avatar that matches the background used by Vidushi Ganeshika, suggesting **Vidushi Ganeshika sent this message**. Both participants appear to have their microphones muted (based on the red microphone icon next to "OnePlus CPH2447" at the bottom).



**Fig:A.8.5.speech received on other device**

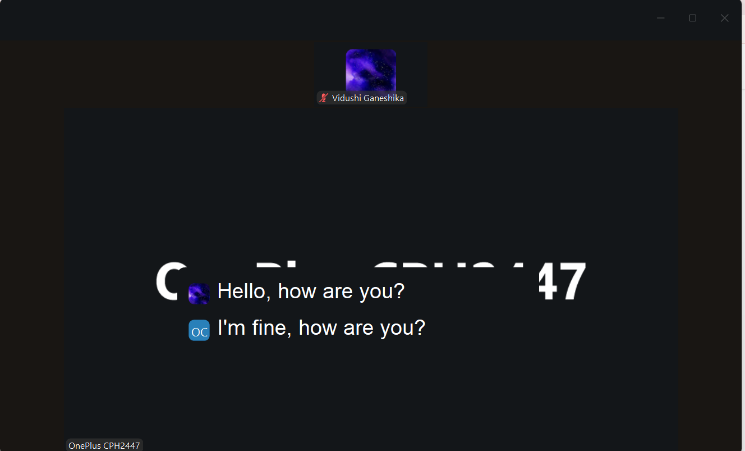
This image displays the screen of a mobile device during an online video call or meeting, likely in full-screen mode.

The background is entirely black, with a small, central square showing a purple, starry virtual background. A chat message is visible at the bottom, reading "Hello, how are you?" and is attributed to "Vidushi Ganeshika" (indicated by the name and the matching avatar). This suggests the participant named Vidushi Ganeshika has sent a chat message. The top left shows a red icon indicating the microphone is muted



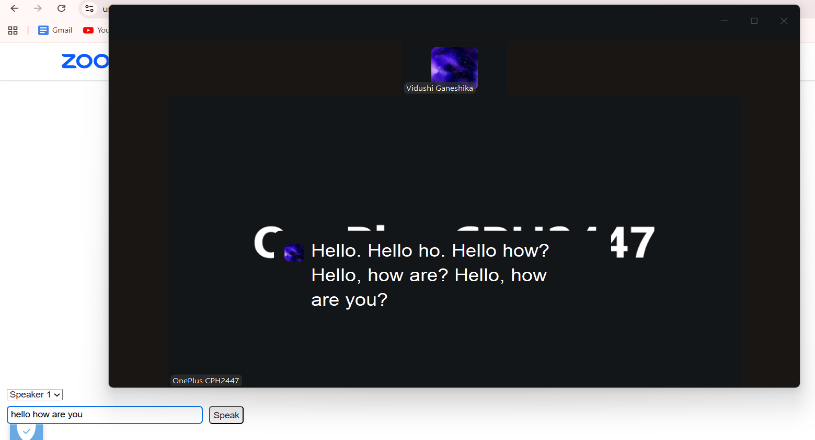
**Fig:A.8.6.speech response replied by other device**

This image shows a mobile screen during an active video conference call, similar to the previous one.The dark screen displays a small, central square with the familiar purple, starry virtual background. A chat message is visible at the bottom that reads, "I'm fine, how are you?" This message appears to be a response to the previous inquiry. The name "Vidushi Ganeshika" is now visible next to a red microphone-slash icon, which indicates that Vidushi Ganeshika is currently muted in the call. The overall presentation is that of a full-screen, mobile-based meeting with a chat overlay



**Fig:A.8.7.speech received on main device**

This image captures a segment of a computer-based video conference screen, showing an ongoing chat exchange.The participant named "Vidushi Ganeshika" is visible in a small tile at the top, using the purple, starry virtual background and is muted. The main screen is dominated by the device name "OnePlus CPH2447" (likely the other participant's unmuted video/display). Overlaid on the screen is a chat history: "Hello, how are you?" (from Vidushi Ganeshika, using the purple avatar) and the reply, "I'm fine, how are you?" (from the other participant, indicated by the "OC" avatar). This shows a brief text conversation occurring within the video meeting interface.

****

**Fig:A.8.8. text in case if the person can’t talk**

This image displays a screen from a video conferencing platform, likely Zoom, showing a text-to-speech or real-time transcription feature in use.The main screen is focused on the participant named **"OnePlus CPH2447"** (whose video is off). The participant **"Vidushi Ganeshika"** is in a small tile at the top with a virtual background. A large block of text on the screen, starting with **"Hello. Hello ho. Hello how?"** represents a **transcript of spoken words**, indicated by the purple avatar next to the text (likely from Vidushi Ganeshika). The bottom left shows a **"Speaker 1"** dropdown and a text box with **"hello how are you"** and a **"Speak"** button, confirming a transcription or text-to-speech functionality is active on the host's screen.



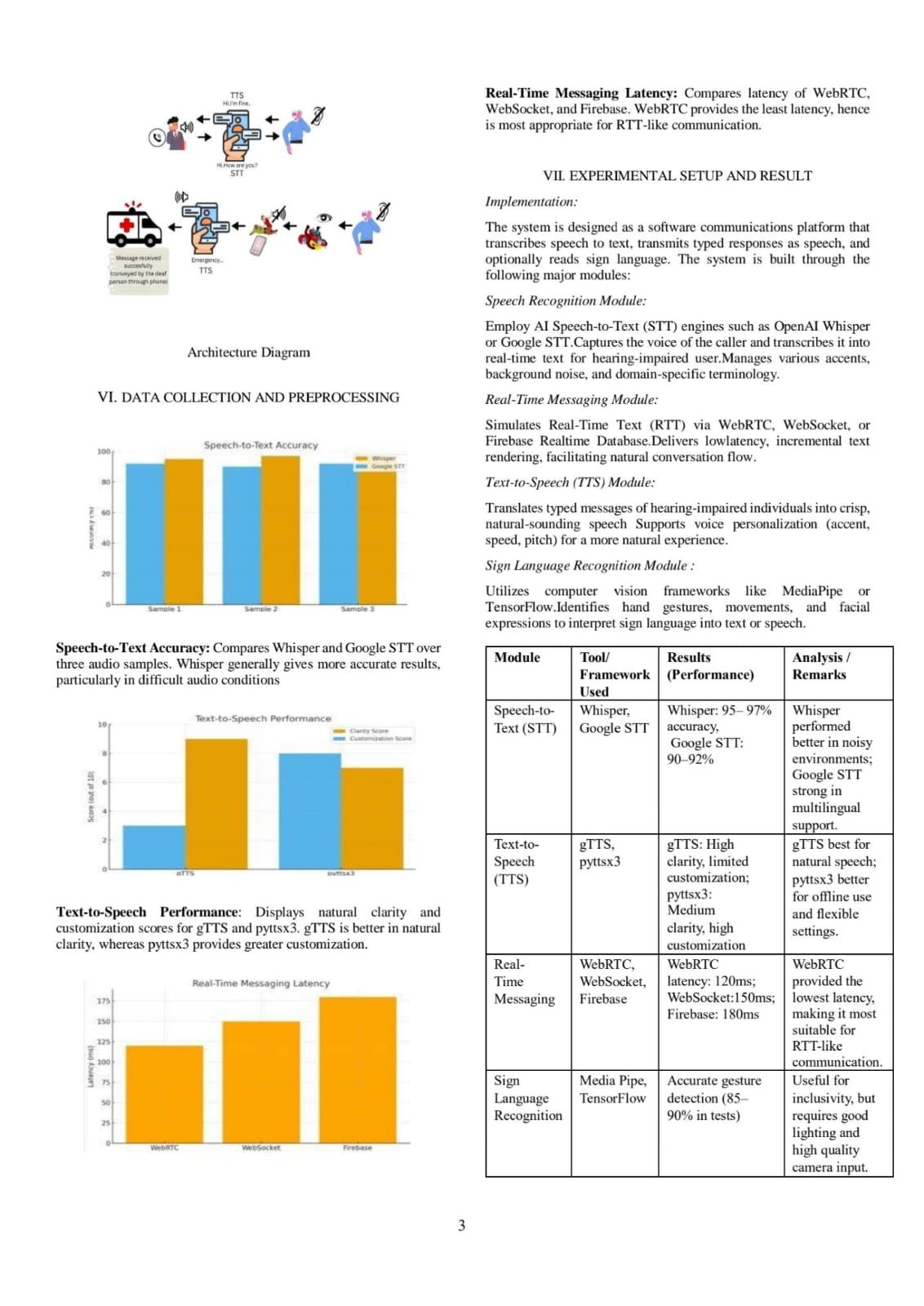
**Fig:A.8.9.text converted to speech and received on other device**

This image displays a full-screen view of a **Zoom meeting** on a mobile device.The top bar confirms the platform with the **"Zoom"** logo and a green shield, indicating security. The **"Leave"** button is prominent in red. The dark background centers on the small video tile of a participant using the **purple, starry virtual background**. A chat message is visible, **"Hello, how are you?"**, sent by the participant **"Vidushi Ganeshika."** The bottom toolbar provides essential controls: **Mute, Start video, Participants, Chat, and Reactions**. The current status shows the participant's mic is muted (indicated by the "Mute" option being available).

****

**A3 PAPER PUBLICATION**

****

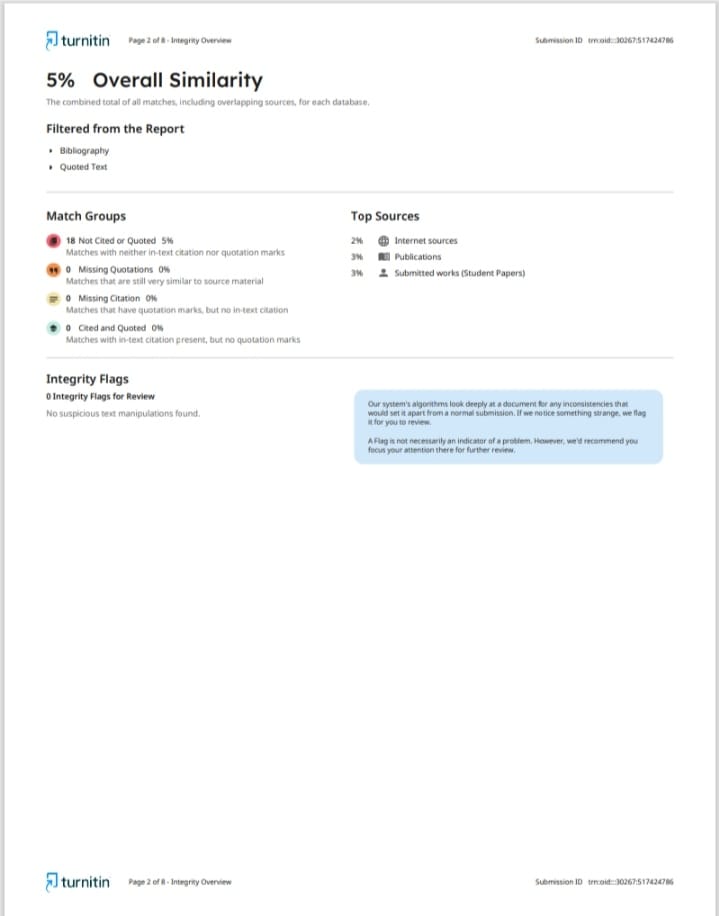
****

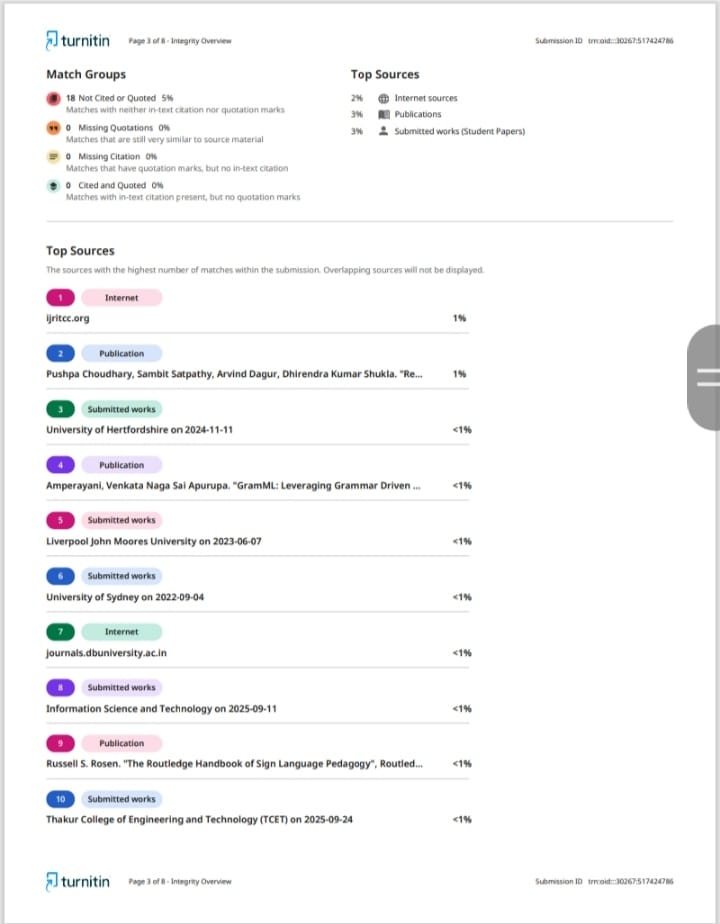
****

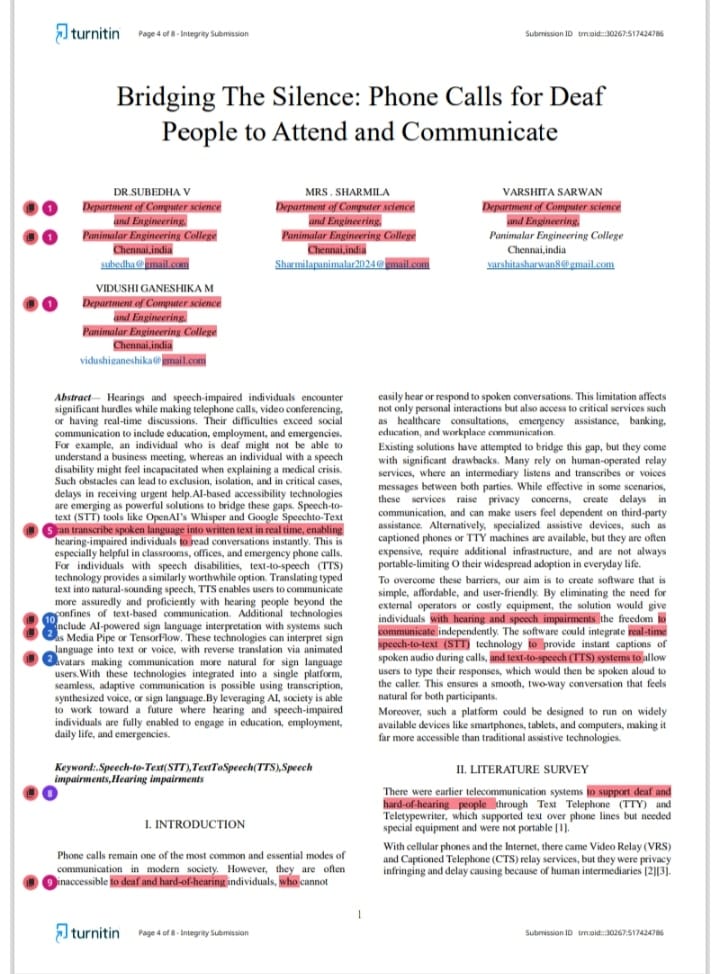
****

**A4. PLAGIARISM REPORT**

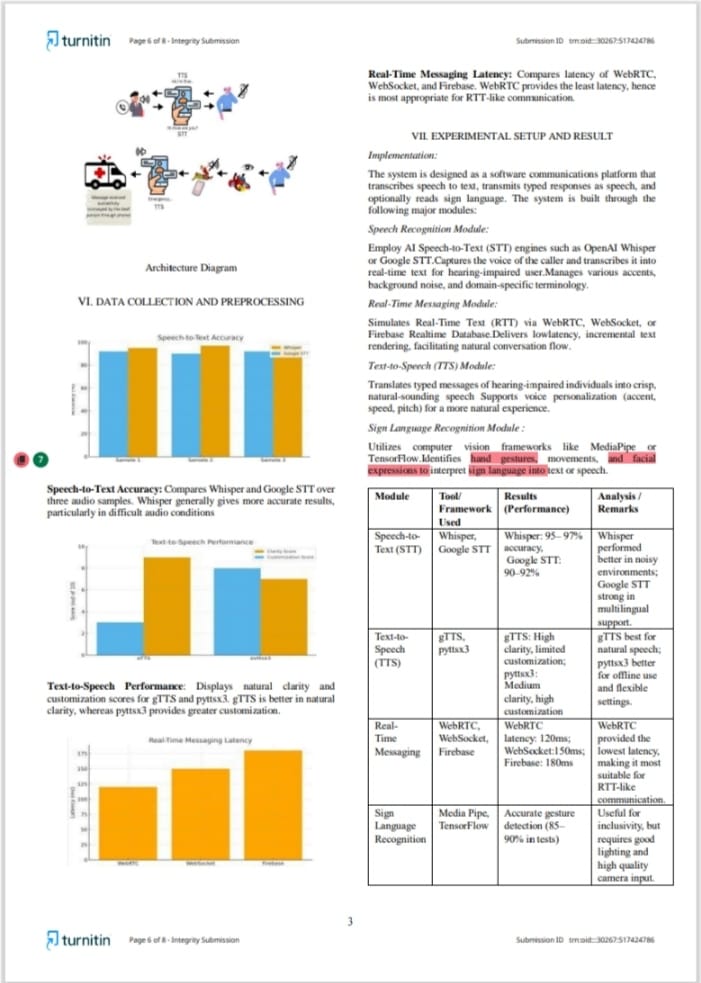


****

****

****

****

****

****

****

**CHAPTER 9**

# REFERENCES

1. Smartphone speech-to-text applications for communication with profoundly deaf patients (PubMed) [PubMed](https://pubmed.ncbi.nlm.nih.gov/26611260/?utm_source=chatgpt.com" \t "_blank)
2. Advancements in Indian Sign Language Recognition Systems: Enhancing Communication and Accessibility for the Deaf and Hearing Impaired (Asian Journal of Electrical Sciences, 2023) [The Research Publication](https://ojs.trp.org.in/index.php/ajes/article/view/4132?utm_source=chatgpt.com" \t "_blank)
3. Indian Sign Language to Text/Speech Translation (IJRASET, 2025) [IJRASET](https://www.ijraset.com/research-paper/indian-sign-language-to-text-speech-translation?utm_source=chatgpt.com" \t "_blank)
4. Towards Real-Time Sign Language Recognition and Translation on Edge Devices (ACM Multimedia) [ACM Digital Library](https://dl.acm.org/doi/abs/10.1145/3581783.3611820?utm_source=chatgpt.com" \t "_blank)
5. Real-Time Sign Language Recognition in Digital Meetings (IJRASET) [IJRASET](https://www.ijraset.com/research-paper/real-time-sign-language-recognition-in-digital-meeting?utm_source=chatgpt.com" \t "_blank)
6. Shruti: An Embedded Text-to-Speech System for Indian Languages (Mukhopadhyay et al.) [ResearchGate](https://www.researchgate.net/publication/3422228_Shruti_An_embedded_text-to-speech_system_for_Indian_languages?utm_source=chatgpt.com" \t "_blank)
7. Advancements in Indian Sign Language Recognition Systems: Enhancing Communication and Accessibility for the Deaf and Hearing Impaired (Asian Journal of Electrical Sciences, 2023) [The Research Publication](https://ojs.trp.org.in/index.php/ajes/article/view/4132?utm_source=chatgpt.com" \t "_blank)
8. Real-Time Gesture Recognition for Sign Language Using Multimodal Techniques (IRJAEH, 2025) [irjaeh.com](https://irjaeh.com/index.php/journal/article/view/736?utm_source=chatgpt.com" \t "_blank)
9. Real-Time Language Translator with Sign Language Recognition: A Multimodal Approach (Kapil et al.) [IJISRT](https://www.ijisrt.com/realtime-language-translator-with-sign-language-recognition-a-multimodal-approach?utm_source=chatgpt.com" \t "_blank)
10. Text-to-Speech (TTS) System for Deafened and Vocally Impaired Persons in Native Language (Tamil, etc.) [SAGE Journals](https://journals.sagepub.com/doi/abs/10.3233/JIFS-231680?utm_source=chatgpt.com" \t "_blank)
11. BhashaBlend: Enabling Multilingual understanding of videos through NLP for Deaf and Hearing Impaired users (IJRASET) [IJRASET](https://www.ijraset.com/research-paper/enabling-multilingual-understanding-of-videos-through-nlp?utm_source=chatgpt.com" \t "_blank)
12. The Role of Existing Technology Text to Speech in the Deaf Disabled Communities Due to Brain Inflammation (Springer) [SpringerLink](https://link.springer.com/chapter/10.1007/978-981-19-4960-9_14?utm_source=chatgpt.com" \t "_blank)
13. SF-Net: Structured Feature Network for Continuous Sign Language Recognition [arXiv](https://arxiv.org/abs/1908.01341?utm_source=chatgpt.com)
14. Multi-View Spatial-Temporal Network for Continuous Sign Language Recognition (Ronghui Li, Lu Meng) [arXiv](https://arxiv.org/abs/2204.08747?utm_source=chatgpt.com)
15. Sign Language Recognition and Translation: A Multi-Modal Approach Using Computer Vision and Natural Language Processing (Jacky Li et al., RANLP 2023) [ACL Anthology](https://aclanthology.org/2023.ranlp-1.71/?utm_source=chatgpt.com)