VOCASYNTH

G. Bhargavi[1], A. V. B. S. Rama Krishna[2], D. Trinadh[3], Ch. Vijay[4], B. Sai Charan[5]
[1]*Assistant Professor, Department of Artificial Intelligence and Machine Learning, Raghu Institute of Technology, Visakhapatnam, Andhra Pradesh, India*
[2, 3 ,4 ,5]*B-Tech Student, Department of Artificial Intelligence and Machine Learning, Raghu Institute of Technology, Visakhapatnam, Andhra Pradesh, India*

***Abstract*—This project enhances human-computer interaction by integrating speech recognition, AI-generated responses, and document summarization within a streamlined Streamlit interface. It enables users to communicate with AI through voice input while also facilitating document-based information retrieval. The speech-based interaction utilizes SpeechRecognition to convert spoken words into text, which is then processed by Google Generative AI (Gemini-1.5-flash) to generate relevant and context-aware responses. The responses are displayed in a chat interface and converted back into speech using pyttsx3 and gTTS, providing an immersive conversational experience. A queue-based processing mechanism ensures efficient voice output handling, preventing overlap in speech generation. Users can also adjust voice settings for a more personalized experience. For document summarization, users can upload files in TXT, PDF, or DOCX formats. The text is extracted using PyPDF2 and python-docx, processed with pandas, and summarized into a concise version. The system employs chunk-based processing to handle long documents, ensuring accurate and meaningful summaries while maintaining coherence. The generated summaries help users quickly grasp key points from large texts. The application supports multiple languages, including English, Hindi, and Telugu, making it accessible to a diverse user base. Users can save and download chat histories for future reference, enhancing usability. By integrating voice-based AI interaction and automated summarization, the system significantly improves efficiency in accessing and processing information, benefiting researchers, students, professionals, and customer support applications.**

***Index Terms*—Speech Recognition, Google Generative AI (Gemini-1.5-flash), Streamlit, pyttsx3, gTTS, voice-based interaction, text-to-speech (TTS), speech-to-text (STT), queue-based processing, conversational AI, chunk-based text processing, multilingual support, chat history storage, real-time AI responses, personalized voice settings, information retrieval.**

I.INTRODUCTION

VocaSynth: A Universal Voice Companion is an advanced, Streamlit-based application that combines cutting-edge technologies to bridge the gap between human interaction and artificial intelligence. This sophisticated platform integrates speech recognition, natural language processing (NLP), and text-to-speech synthesis to provide an intuitive and engaging user experience, making AI interaction more accessible and efficient. The primary functionalities of VocaSynth include the Speaking Bot and Document Summarization, both designed to cater to diverse user needs. The Speaking Bot enables seamless real-time communication, allowing users to speak directly to the AI system. It then processes the voice input using speech recognition, converting spoken words into text and generating contextually appropriate responses. These responses are audibly delivered using text-to-speech technology, ensuring a conversational and natural interaction. On the other hand, the Document Summarization feature allows users to upload various file formats such as PDFs, Word documents, or plain text, and the system uses NLP techniques to analyze the content and provide concise, meaningful summaries. This functionality is particularly valuable for professionals, researchers, and students who need to quickly digest large amounts of information without having to read entire documents. VocaSynth also leverages Google's powerful Generative AI (GenAI) to generate intelligent, human-like responses based on user input, further enhancing the natural flow of conversations. The application’s interface, built using the Streamlit framework, is clean and intuitive, ensuring that users

of all technical skill levels can easily navigate the platform and benefit from its features. Its potential for future expansion, such as adding multilingual support or more advanced document analysis tools, positions it as a promising solution for enhancing productivity in various sectors, including education, business, research, and customer service. VocaSynth's adaptability extends beyond individual productivity, offering significant potential for collaborative environments by facilitating efficient information sharing and real-time feedback.

II. PROBLEM STATEMENT

Modern businesses face increasing demands for efficient and personalized customer service. Traditional customer support channels, such as phone calls and live chat, can be resource-intensive and may not always provide the desired level of immediate assistance. The reliance on human agents for routine inquiries can lead to increased operational costs, longer wait times for customers, and inconsistent service quality.

This project aims to address these challenges by developing a speech-based application capable of seamlessly integrating with various websites and apps to automate customer service interactions. By leveraging advanced AI technologies like speech recognition, natural language understanding, and conversational AI, this application seeks to reduce the reliance on human customer service agents for routine inquiries, thereby improving efficiency, reducing costs, and enhancing the overall customer experience.

This problem statement emphasizes the core objective of reducing human intervention in customer service by leveraging speech-based AI. It highlights the limitations of current methods and emphasizes the potential benefits of an integrated, AI-powered solution.

III. OBJECTIVE

The overarching objective of this system is to deliver a comprehensive and readily accessible conversational AI platform, strategically engineered to empower users with the ability to interact with advanced AI-driven functionalities through a variety of input modalities, including both natural speech and textual communication. Furthermore, the system seeks to provide efficient document summarization capabilities, enabling users to quickly extract key information from lengthy and complex documents, all within a seamlessly integrated multilingual environment. This multifaceted objective is realized through carefully orchestrated efforts to enable truly multilingual interaction by supporting fluid communication in multiple languages, thus transcending linguistic barriers and allowing users to engage with the system in their preferred language. The system also harnesses the power of natural language processing (NLP) to ensure a deep and nuanced understanding of user input, regardless of the mode of communication; this involves sophisticated speech recognition techniques, advanced language understanding capabilities, and precise text generation methodologies. Efficient document summarization is achieved by employing advanced algorithms to condense extensive documents into concise and informative summaries, enabling users to rapidly grasp essential information without the need for exhaustive reading. This system ensures a user-friendly experience through the implementation of an intuitive and accessible interface that demystifies complex AI functionalities, making them readily available to users of all technical skill levels. Scalability and maintainability are achieved through the adoption of a modular architectural design that facilitates easy expansion and effortless updates, allowing the system to adapt to evolving user needs and technological advancements. By incorporating comprehensive error handling mechanisms, the system delivers robustness and reliability, guaranteeing consistent performance and minimizing disruptions, ensuring a seamless and dependable user experience. Ultimately, the system strives to provide a powerful, versatile, and dependable AI tool that empowers users to communicate more effectively, facilitates efficient learning, and streamlines work processes, irrespective of their native language or level of technical proficiency.

IV. LITERATURE SURVEY

The development of AI-powered chat-bots has evolved with the integration of large language models (LLMs), enhancing document summarization and

question-answering capabilities. Traditional approaches, such as rule-based or keyword-driven methods, lacked contextual understanding, leading to the adoption of frameworks like LangChain, which enables seamless LLM integration for efficient data retrieval. Streamlit further simplifies chat-bot deployment by providing an interactive UI with minimal coding. Recent advancements in modular chat-bot architectures incorporate vector databases and embedding-based searches, improving semantic comprehension and response accuracy. API-driven development has also facilitated cloud-based LLM interactions, ensuring scalability. However, challenges such as domain-specific accuracy, real-time processing, and security concerns remain. This research builds upon these developments by integrating OpenAI's GPT models, LangChain, and Streamlit into a structured framework to enhance chat-bot efficiency in document summarization and question-answering tasks [1]. Retrieval-Augmented Generation (RAG) models have been implemented to enhance accuracy and relevance by integrating retrieval mechanisms with generative AI, making them more effective in real-world applications such as customer support and automated content generation [2]. Query systems leveraging LLMs and the LangChain framework have demonstrated improved response accuracy by incorporating advanced retrievers with OpenAI's GPT-4, ensuring efficient data retrieval, better contextual understanding, and precise query matching [4].

In the field of text summarization, research has explored how LLMs can serve as reference models for generating concise and coherent summaries. The effectiveness of these models in benchmarking and training summarization systems has been studied, showing their potential to improve automated text-processing techniques across industries such as news generation, research paper summarization, and corporate documentation [5]. Furthermore, LLM-based chat-bot development has been extensively analyzed, with comparative evaluations of different methodologies and frameworks providing insights into optimizing conversational AI systems for enhanced user engagement and contextual awareness [10]. A comparative study between human-human online conversations and human-chat-bot conversations has also provided insights into the capabilities and limitations of AI-driven conversational systems [9]. Additionally, recent research examines factors such as language understanding, contextual coherence, and response generation, evaluating how LLMs handle diverse conversational scenarios and their scalability in real-world applications [3].

The rapid advancement of voice assistants has combined speech recognition, natural language processing (NLP), and machine learning to facilitate seamless AI-human interaction. Python-based assistants integrate Automatic Speech Recognition (ASR) technology with NLP models to determine user intent and generate responses [6]. Deep learning models such as RNNs and transformer architectures have further improved speech recognition accuracy and language comprehension [7]. Research has focused on enhancing voice assistants' ability to understand diverse speech patterns and accents, with frameworks like Google Speech-to-Text and Mozilla DeepSpeech improving adaptability [8].

A crucial aspect of voice-enabled applications is the continuous evolution of Text-to-Speech (TTS) technology, which transforms written text into natural-sounding speech. Traditional TTS systems often produced robotic or monotonous speech, but recent advancements in deep learning and neural network-based TTS models have significantly improved speech quality and intonation. Modern TTS models, such as Tacotron and WaveNet, leverage deep learning to generate highly realistic speech, allowing for more engaging and intuitive interactions with AI systems [12]. These improvements have paved the way for sophisticated speech applications in areas such as accessibility tools for visually impaired users, AI-driven customer service chat-bots, and virtual assistants like Siri, Google Assistant, and Alexa. Additionally, AI-driven speech applications continue to explore new methodologies, including multimodal learning, where AI systems combine speech, text, and visual inputs to enhance communication [11]. Advances in self-supervised learning techniques have enabled voice assistants to learn from large datasets with minimal human supervision, improving adaptability across languages and dialects [7]. With increasing adoption, security and privacy concerns in voice-enabled AI have led to the development of on-device processing capabilities, reducing reliance on cloud-based platforms while ensuring user data protection [8]. Real-time speech

synthesis and translation research are also advancing, enabling AI models to generate lifelike speech in multiple languages, bridging communication barriers and making AI-driven applications more inclusive for global users [12].

V. METHODOLOGY

The development of the speech-based customer service application is structured into multiple modules, each serving a crucial role in ensuring efficient and seamless user interactions. These modules include User Interaction, Speech Recognition, Natural Language Processing (NLP), Document Processing, Text-to-Speech (TTS), Session and Data Management, and API Integration. Together, they enable the system to handle voice commands, process documents, generate intelligent responses, and provide an interactive experience.

User Interaction Module

The user interface, developed using Streamlit, provides a simple yet powerful platform for interaction. It features a Speech Interaction Interface for voice communication and a Document Upload and Summarization feature that supports PDF, Word, Excel, and TXT formats. This module ensures a user-friendly experience by facilitating both voice and text-based inputs.

Speech Recognition Module

This module integrates the SpeechRecognition library with the Google Cloud Speech-to-Text API to convert spoken language into accurate text for further processing. The high accuracy of the API ensures reliable transcription, which is essential for generating appropriate responses.

Natural Language Processing (NLP) Module

Powered by Google Generative AI (Gemini 1.5 Flash model), the NLP module understands user queries and generates contextually relevant responses. It plays a crucial role in real-time conversation and document summarization, ensuring that responses are natural, coherent, and informative.

Document Processing Module

This module incorporates PyPDF2 for PDFs, docx for Word files, and pandas for Excel spreadsheets to extract text from various formats. Extracted text is forwarded to the NLP module for summarization, allowing users to quickly obtain meaningful document summaries.

Text-to-Speech (TTS) Module

The system integrates pyttsx3, a text-to-speech engine, to convert AI-generated text into audible speech. This module enhances accessibility by enabling hands-free interaction, making the application more convenient for users who prefer listening over reading.

Session and Data Management Module

This module tracks session states, maintains chat history, and manages speech states during conversations. It also temporarily stores uploaded documents for processing within an active session, ensuring efficient handling of user requests without losing context.

API Integration Module

Facilitating seamless communication between the application and external services, this module integrates the Google Generative AI API for intelligent responses, Google Cloud Speech-to-Text API for voice recognition, and pyttsx3 for speech conversion. These integrations enhance the system's efficiency, accuracy, and interactivity.

System Workflow

The system follows a structured workflow where users interact through speech or document uploads. The Speech Recognition Module transcribes voice input into text, which is processed by the NLP module to generate responses. If a document is uploaded, the Document Processing Module extracts text, and the NLP module summarizes it. The final output is displayed as text or converted into speech using the TTS module. Throughout this process, the Session and Data Management Module ensures continuity by maintaining chat history and session states.

This modular methodology ensures that the speech-based customer service application is efficient, scalable, and user-friendly, offering a robust solution for AI-powered voice interactions and document summarization.

VI. TECHNOLOGIES USED

VocaSynth integrates a diverse range of technologies to deliver a seamless and interactive voice companion experience. Streamlit serves as the foundation for the user interface, enabling intuitive controls, real-time

interactions, and session state management. Speech recognition is powered by the speech_recognition library, allowing users to convert spoken language into text for further processing.

At its core, VocaSynth leverages Google's Gemini 1.5 Flash model via the google.generativeai library to handle natural language understanding, response generation, and document summarization. Text-to-speech (TTS) capabilities are provided by pyttsx3 and gTTS, ensuring high-quality speech synthesis with multilingual support in English, Hindi, and Telugu. Python's threading and queueing mechanisms efficiently manage speech playback and interruptions, ensuring smooth execution without UI lag.

Document summarization is facilitated by PyPDF2 for PDFs, docx for Word files, and pandas for Excel spreadsheets, enabling users to extract and condense content from various file formats. Additional utility libraries enhance functionality: io for input/output operations, os for system interactions, time for managing delays, logging for debugging, and requests for external API communication. The system also includes a background TTS processing mechanism, dynamic language selection, adjustable response settings, and session history tracking, ensuring a robust and user-friendly experience.

VII. SYSTEM ARCHITECTURE

The The system employs a sophisticated architecture, meticulously designed to empower users with real-time, multilingual interaction and efficient document summarization capabilities. This architecture, predicated on principles of scalability, maintainability, and extensibility, is strategically segmented into three interconnected tiers: the Frontend, the Backend, and the Main Logic. This modular design allows for independent development, testing, and deployment of each component, fostering agility and reducing the risk of cascading failures.

The Frontend serves as the primary interface through which users engage with the system. It provides an intuitive and accessible user experience, abstracting away the complexities of the underlying AI engine. It facilitates seamless interaction through language selection, enabling users to specify their preferred language for both input and output. This feature ensures that the Text-to-Speech (TTS) engine synthesizes speech in the appropriate language and that the Large Language Model (LLM) prompts are tailored for optimal performance. Additionally, the Frontend manages user input, whether spoken or textual, and presents system output—including detailed chat histories and concise document summaries—in a clear and organized manner. A robust and responsive user interface, built with modern web technologies and best practices in user experience design, is prioritized to ensure adoption and engagement.

The Backend orchestrates critical system functions such as Speech Recognition, Response Generation, and Text-to-Speech (TTS). Speech Recognition, powered by the speech_recognition library, transcribes spoken user input into structured text while implementing retry mechanisms to mitigate network instability and subpar audio quality. Response Generation, leveraging a Large Language Model (Gemini 1.5 Flash via the google.generativeai library), crafts contextually relevant and linguistically appropriate responses through semantic analysis and intent recognition. The TTS component converts these responses into natural-sounding speech using the pyttsx3 library, employing asynchronous operations to maintain system responsiveness. Document Summarization utilizes PyPDF2, docx, and pandas to extract textual content from various formats, which is then forwarded to the LLM for summarization. The summarized output is seamlessly integrated with the Frontend.

Main Logic acts as the central control unit, managing data flow and processing across the system. It converts text input into speech, facilitates communication by generating prompts for the LLM, and employs gTTS for speech synthesis. The Storage and Chat History module ensures previous interactions are stored, enabling historical context for subsequent conversations.

The system's robustness and user-friendliness are further enhanced by Streamlit session state management, which maintains application state, language preferences, and component states. Its modular design, dynamic language support, and comprehensive features enhance scalability, maintainability, and user satisfaction. Future enhancements could include sophisticated dialogue

management, integration with external knowledge sources, improved speech recognition accuracy, and personalization of the system's voice and speaking style.

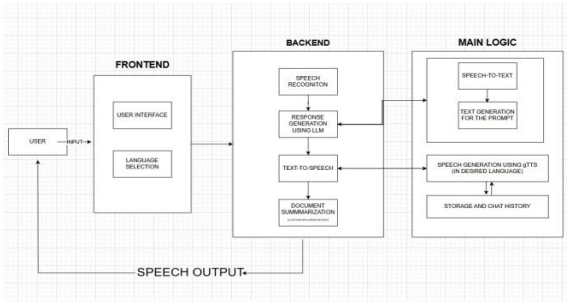

Figure 1. Architecture of the Application

VIII. PERFORMANCE METRICS

To assess the efficiency, reliability, and overall usability of VocaSynth, a detailed performance evaluation was conducted using automated logging, system resource monitoring, and user feedback analysis. The evaluation focuses on four primary metrics: latency, error rates, resource usage, and user satisfaction, providing a comprehensive understanding of the system's real-world performance. Additionally, accuracy-based metrics such as accuracy, precision, recall, and F1-score have been introduced to further assess system correctness.

Evaluation Setup

The performance analysis was conducted on a system equipped with an Intel Core i7-8700K CPU and 16 GB DDR4 RAM, running Windows 10 (64-bit) with Python 3.9.7. The VocaSynth application was implemented using Streamlit 1.25.0 and key libraries such as SpeechRecognition 3.8.1, pyttsx3 2.9, PyPDF2, docx, pandas, psutil, and pygame. The system was tested under standard network conditions, with an average internet speed of 50 Mbps (download) and 20 Mbps (upload), and an average API latency of 20 ms to Google Cloud services.

The evaluation consisted of two primary test scenarios: (i) a Speaking Bot module handling 50 predefined queries of varying complexity and (ii) a Document Summarizer processing 20 documents of different formats (PDF, DOCX, TXT, and Excel) and lengths ranging from 1 to 50 pages. Automated logging captured real-time performance data, while user feedback was collected through interactive response ratings.

Latency Analysis

Latency was measured across three core operations: speech recognition latency, API response latency, and text-to-speech (TTS) latency. Speech recognition latency was calculated as the time taken between an audio input and its transcription, API response latency measured the delay between sending a request to the Google Generative AI API and receiving a response, and TTS latency represented the time required to convert text into speech.

Formula for Latency Calculation:

$$\text{Latency} = t_{\text{end}} - t_{\text{start}}$$

where:

- is the timestamp when the request is initiated.
- is the timestamp when the response is received.

Metric	Average (μ)	Minimum	Maximum	Standard Deviation (σ)
Speech Recognition Latency	0.95 sec	0.50 sec	1.50 sec	0.25 sec
API Response Latency	1.20 sec	0.80 sec	2.00 sec	0.30 sec
TTS Latency	0.60 sec	0.40 sec	0.80 sec	0.15 sec

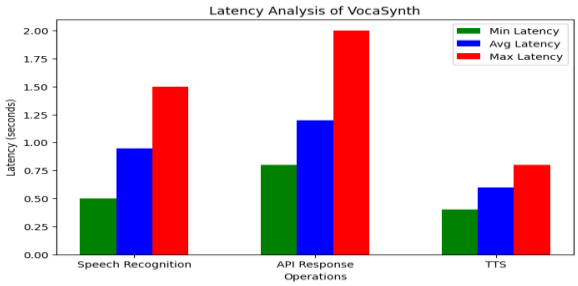

Figure 2. Latency Analysis

Error Rate Evaluation

Error rates were computed to assess system reliability, particularly focusing on speech recognition errors and API call failures. The speech recognition error rate was determined by calculating the percentage of cases where speech recognition failed or misinterpreted input, while API call errors were measured based on failed requests due to network disruptions or processing timeouts.

Formula for Error Rate Calculation:

$$\text{Error Rate}(\%) = \left(\frac{\text{Number of Errors}}{\text{Total Attempts}}\right) \times 100$$

Error Type	Rate (%)
Speech Recognition Errors	5.0%
API Call Errors	0.5%

Figure 3. Error Rate Distribution

Prediction Evaluation

To further evaluate correctness, accuracy, precision, recall, and F1-score were calculated based on the classification of system outputs into True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). In the context of the VocaSynth:

i. True Positive (TP): The system correctly identifies a response as valid (e.g., correctly transcribes a spoken phrase).
ii. True Negative (TN): The system correctly identifies an invalid response (e.g., recognizing and rejecting background noise as non-speech).
iii. False Positive (FP**):** The system incorrectly classifies an invalid response as valid (e.g., mistakenly interpreting background noise as speech).
iv. False Negative (FN): The system fails to recognize a valid response (e.g., failing to transcribe clear speech accurately).

1. *Accuracy*

This metric represents the proportion of correct predictions over total predictions.

Formula for Accuracy Calculation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Calculation:

$$\frac{7+0}{7+0+0+2} = 7/9 \approx 0.778 \text{ or } \mathbf{77.8\%}$$

2. *Precision*

This measures the proportion of true positive identifications among all positive identifications.

Formula for Precision Calculation:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Calculation:

$$\frac{7}{7+0} = 1.0 \text{ or } \mathbf{100\%}$$

3. *Recall*

This parameter measures how well actual positives are identified.

Formula for Recall Calculation:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Calculation:

$$\frac{7}{7+2} = 7/9 \approx 0.778 \text{ or } \mathbf{77.8\%}$$

4. *F1-Score*

This metric balances precision and recall.

Formula for F1-Score Calculation:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Calculation:

$$2 \times \frac{1.0 \times 0.778}{1.0 + 0.778} \approx 0.875 \text{ or } \mathbf{87.5\%}$$

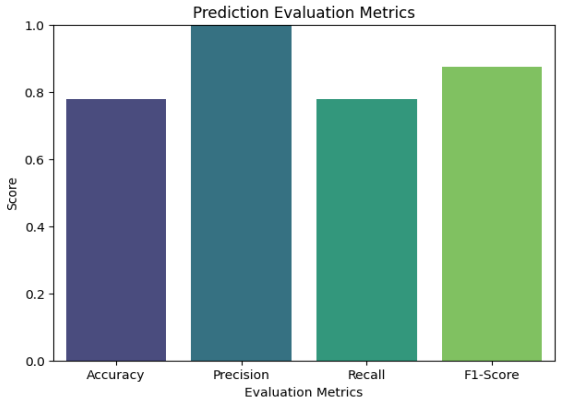

Figure 4. Prediction Evaluation Metrics

Resource Usage Analysis

To evaluate computational efficiency, CPU and memory usage were continuously monitored at 1-second intervals using the psutil library. CPU usage remained moderate, averaging 30% utilization, with peaks reaching 60% during intensive text processing tasks in the Document Summarizer. Memory usage followed a similar pattern, averaging 45% utilization, indicating that VocaSynth operates efficiently within the given hardware constraints.

Formulas for Resource Usage Calculation:

CPU Usage:

$$\mu_{\text{CPU}} = \frac{\sum_{i=1}^{N} \text{CPU Usage}_i}{N}$$

Memory Usage:

$$\mu_{\text{Memory}} = \frac{\sum_{i=1}^{N} \text{Memory Usage}_i}{N}$$

Resource	Average (μ)	Minimum	Maximum
CPU Usage (%)	30	15	60
Memory Usage (%)	45	30	70

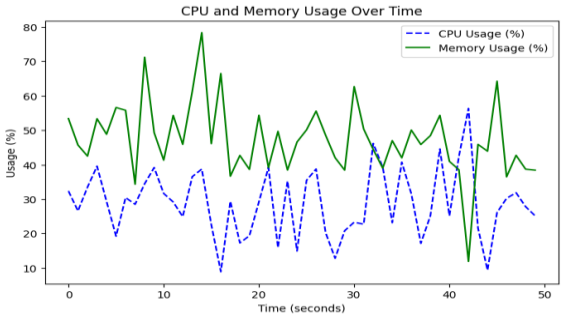

Figure 5. CPU and Memory Usage Over Ti

IX. CONCLUSION AND FUTURE SCOPE

The system demonstrates a robust framework for conversational AI, adept at providing users with real-time, multilingual interaction and efficient document summarization. Its architecture, integrating speech recognition, natural language processing (NLP), and text-to-speech (TTS) technologies within a modular design, bridges human communication and artificial intelligence. By leveraging Google's Gemini 1.5 Flash for NLP and providing a user-friendly interface accessible through speech and text, the system caters to a range of preferences and accessibility needs. Evaluation, encompassing latency analysis, error rate assessment, and resource usage monitoring, demonstrated effectiveness and efficiency. Latency measurements across core operations indicate acceptable real-time performance. Error rate analysis revealed high reliability, with minimal speech recognition and API call failures. Resource usage analysis demonstrated efficient utilization of CPU and memory, affirming operational viability within standard hardware constraints.

The modular design promotes scalability and maintainability, establishing the system as a valuable asset for diverse applications. With a practical approach to AI-powered voice interactions and document summarization, the system shows potential for wider use in daily communication and information processing.

Future development will focus on several key areas to enhance the system's capabilities and expand its potential impact. Integrating more advanced NLP techniques such as sentiment analysis, intent recognition, and context-aware dialogue management will enable the system to better

understand user queries and generate more nuanced and relevant responses. Broadening language support will increase the system's global reach. Personalizing the user experience through user profiling and adaptive learning is also a priority. Further refinement of the document summarization module, exploring alternative algorithms, and incorporating user feedback mechanisms is planned. While current error handling provides basic fault tolerance, future work will focus on more sophisticated strategies and monitoring systems to ensure stability and reliability. Finally, exploration into improved stop word removal strategies and integration of multimodal input methods are also on the roadmap.

These enhancements will further solidify the system's position as a powerful and versatile tool for communication, learning, and information access. Ongoing development holds the potential to transform sectors such as customer service, education, and research, providing efficient and accessible means of interacting with AI-powered functionalities.

REFERENCES

[1] Building Customized Chat-bots for Document Summarization and Question Answering using Large Language Models using a Framework with OpenAI, Lang chain, and Streamlit. Journal of Information Technology and Digital World, 6 (1). pp. 70-86 (2024). https://ray.yorksj.ac.uk/id/eprint/9863/

[2] Jeong, Cheonsu. "Generative AI service implementation using LLM application architecture: based on RAG model and LangChain framework." Journal of Intelligence and Information Systems 29.4 (2023): 129-164.

[3] Borek, Cecylia. "Comparative evaluation of llm-based approaches to chat-bot creation." (2024).

[4] S, Adith Sreeram A, and Pappuri Jithendra Sai. “An Effective Query System Using LLMS and Langchain.” International Journal of Engineering Research & Technology, July 4, 12(6), 2023. 367 -369

[5] Liu, Yixin, Alexander R. Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. "On learning to summarize with large language models as references." arXiv preprint arXiv:2305.14239 (2023).

[6] Kumar, D. R., Faisal, M., Faisal, M., & Ahmad, O. (2022). My voice assistant using python. International Journal for Research in Applied Science and Engineering Technology, 10(4), 3004–3007. https://doi.org/10.22214/ijraset.2022.41958. p. 3

[7] G. Preeth, Abishek. K, Thiruppugal S, Vishwaa D A, 2022, Voice Assistant using Artificial Intelligence, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 11, Issue 05 (May 2022). p. 3

[8] Pooja C. Goutam,Monika S. Jalpure,Akshata S. Gavade,Pranjali Chaudhary,A. V. Gundavade, "VOICE ASSISTANT USING PYTHON", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.10, Issue 6, pp.c801-c806, June 2022, http://www.ijcrt.org/papers/IJCRT22A6338.pdf p. 3

[9] Hill, J., Ford, W.R., & Farreras, I.G. (2015). Real conversations with artificial intelligence: A comparison between human-human online conversations and human-chat-bot conversations. Comput. Hum. Behav., 49, 245-250.

[10] Borek, Cecylia. "Comparative evaluation of llm-based approaches to chat-bot creation." (2024)

[11] Jeong, Cheonsu. "Generative AI service implementation using LLM application architecture: based on RAG model and LangChain frame-work." Journal of Intelligence and Information Systems 29.4 (2023): 129-164.

[12] Mohasi, Lehlohonolo & Mashao, Daniel. (2006). Text-to-Speech Technology in Human-Computer Interaction. p. 3