STM JOURNALS

Current Trends in
Information Technology

https://journals.stmjournals.com/ctit

ISSN: 2249-4707 (Online)
ISSN: 2348-7895 (Print)
Volume 14, Issue 2, 2024
May–August
DOI (Journal): 10.37591/CTIT

Review CTIT

Jarvis the Voice Assistant for Desktop

Sannidhi Padma Prasuna[1,*], Godina Geethika[1],
Sai Sarayu V.[1], Gayathri J.[2]

Abstract

This study presents the development and execution of a cutting-edge multilingual voice partner using JavaScript, HTML, CSS, and Python for the frontend and backend, independently. The voice accomplice is planned to appreciate and address client inquiries in various lingos, making it a significant gadget for people of all ages and foundations. The front end of the voice associate is created using JavaScript, HTML, and CSS, which give a simple to utilize point of interaction and assurance cross-stage closeness. The voice partner's multilingual limits are achieved utilizing advanced ordinary language taking care of methodologies, including language acknowledgment and understanding. The voice helper's execution in WhatsApp and various applications is achieved utilizing APIs and web snares, which consider continuous correspondence between the voice accomplice and the application. The voice accomplice can be gotten to through a direct visit interface, enabling clients to team up with the voice partner using ordinary language requests. The voice partner's responses can be passed on through message, voice, or intuitive media messages, dependent upon the client's tendencies and the capacities of the application. With the rising popularity of illuminating applications as a fundamental technique for correspondence, the voice accomplice's blend into these stages gives a supportive and open way for clients to interact with the voice of the right hand.

Keywords: Voice assistant, multilingual coordination, regular language handling, WhatsApp, client experience

INTRODUCTION

Introducing Jarvis, our own voice assistant, which draws inspiration from the cutting-edge artificial intelligence (AI) portrayed in the heroic technology realm. Jarvis is made to help you with a lot of things, like controlling and organizing your smart devices and giving you the most recent news and weather information. Jarvis makes our life easier and more connected by using its speech interaction, which is easy to use. Jarvis remembers our preferences and provides personalized advice and reminders. Jarvis is here to make every day easier and more productive, whether you need help with professional or personal matters.

***Author for Correspondence**
Sannidhi Padma Prasuna
E-mail: padmaprasunas@gmail.com

[1]Student, School of Information Science SOIS, Presidency University, Ittagallpura, Karnataka, India
[2]Assistant Professor, School of Information Science SOIS, Presidency University, Ittagallpura, Karnataka, India

Received Date: May 31, 2024
Accepted Date: June 05, 2024
Published Date: July 05, 2024

Citation: Sannidhi Padma Prasuna, Godina Geethika, Sai Sarayu V., Gayathri J. Jarvis the Voice Assistant for Desktop. Current Trends in Information Technology. 2024; 14(2): 28–39p.

Real-time Information Updates

Keep up to date on the most recent events, traffic reports, and weather predictions. To ensure that you are always ready, whether it is choosing weather-appropriate clothing or figuring out the ideal commute route, Jarvis can offer comprehensive reports customized to your area and tastes. *Entertainment and leisure:* Jarvis can arrange movie evenings, play music, or even recommend a good book depending on your preferences. Jarvis transforms your living area into an entertainment center that is precisely tailored to your tastes, giving you access to streaming services and digital libraries.

Facilitation of Communication

Construct video conferences, make calls, and send messages on demand. With Jarvis, staying in touch with friends, family, and coworkers is easier than ever thanks to its interface with your contacts and calendar.

Academic and Learning Support

Jarvis can help with anything from a fast fact check to language translation to a thorough analysis of a challenging subject. Using voice commands, it is similar to having immediate access to a huge library of knowledge.

Safety and Secrecy

Maintaining your privacy is crucial. To safeguard your data and personal information, Jarvis is developed with the best security features available. In addition to its many benefits, Jarvis offers peace of mind by staying ahead of security concerns thanks to regular updates.

METHODOLOGY FRAMEWORK

Components

Speech Recognition

This module uses the user's microphone to translate spoken audio into text. The Google Speech-to-Text API (https://cloud.google.com/speech-to-text) and Speech Recognition (https://pypi.org/project/SpeechRecognition/) are two well-known libraries.

Natural Language Processing (NLP)

This module reads user content, determines its meaning, and extracts pertinent data.

Actionable Logic

This module uses its interpretation of the user's intent to guide its actions.

This Could Entail

Obtaining information via web scraping or API calls (weather, news); manipulating the gadget (starting apps, playing music, etc.) tasks related to text processing (translation, summarization).

Text-to-Speech (TTS)

The user can hear this module's conversion of generated text responses back into speech. You can use libraries such as pyttsx3 (https://pyttsx3.readthedocs.io/).

Framework

Wake Word Detection

The helper watches for the activation of a certain word or phrase (such as “Hey Jarvis”).

Voice Capture and Recognition

When this feature is turned on, the assistant uses the microphone to record the user's spoken commands and then uses voice recognition to turn them into text.

Intent Understanding (NLU)

The user's intent (e.g., “What's the weather today?”) is understood by the NLU module through text processing.

Action Execution

The logic module carries out the desired action in accordance with the intent.

This Could Entail

Obtaining information via the web (weather API); operating the apparatus (music player); replying with a text message (news summary). Voice Output (Voice): The action module's generated response is transformed back into voice for the user by the TTS module.

PROPOSOED METHODOLOGY

A visual representation detailing the process flow of voice assistance systems from user input to final response as shown in Figure 1.

Describe the Functionalities

Determine the essential features you desire. Your helper to carry out (music playback, weather updates, etc.).

Choose Libraries

For each component (text-to-speech, NLU, and speech recognition), choose the suitable Python libraries.

Create Modules

Create separate modules for every feature (such as speech recognition, wake word detection, action logic, NLU, and TTS).

Integrate Modules

Assemble the modules into a single framework, with a main loop that responds to commands, processes wake words, and continuously listens for them.

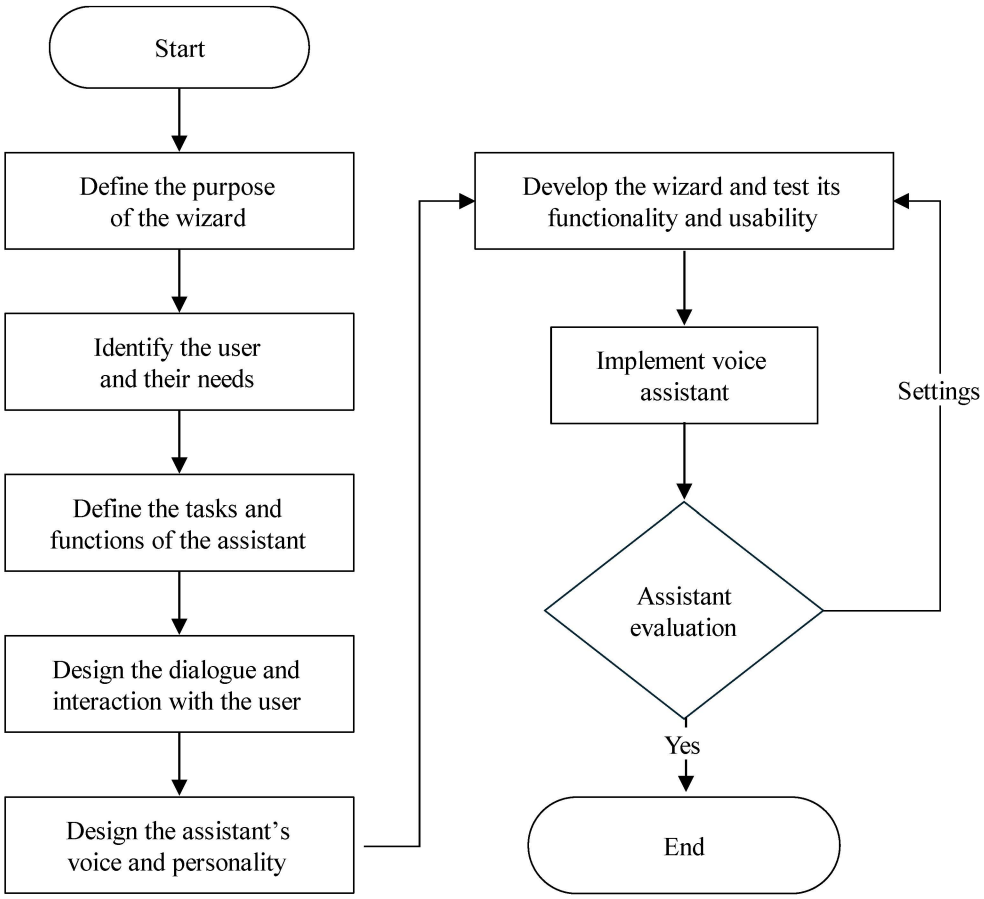

Figure 1. Flowchart voice assistance.

Testing and Refinement

Use a variety of user inquiries to test the assistant, pinpoint areas action logic modules.

IMPLEMENTATION

These are some of the important libraries we used in our Voice Assistant:

Speech Recognition

Uses the microphone on the computer to listen for human input; uses Google Speech Recognition or other models to identify the speech; and returns the speech that was identified as text.

Use import speech recognition as SR to include the speech recognition library.

Make an instance of a recognizer: Using r=sr.Recognizer(), create a Recognizer object.

Listen with Microphone: To access the microphone and record audio, use sr.Microphone() as source.

Record Audio

To record user speech, call audio =re.listens(source). Use command =r.recognize google(audio) to recognize speech. Lower() to use Google Speech Recognition to convert the recorded audio to text, then convert it to lowercase for simpler processing.

Error Handling

Incorporate exception handling for errors such as sr.Request Error for connection problems or sr.Unknown Value Error when speech recognition fails. It is possible to prompt

Text-Speech Recognition

Pyttsx3

Text to speech conversion is done for audible responses.

Voice Assistant function: Reads text input, and plays the converted voice through the computer's speakers. Steps taken:

Import

Use import pyttsx3 to include the pyttsx3 library. *Start the Engine:* Using engine =pyttsx3.init(), create an instance of the text-to-speech engine system.

Predetermined Speech

Employ the engine. To specify the text, you want the helper to read, type say(text).

Act Out a Speech

Dial the engine. To translate the text to speech and play it through the speakers, use run And Wait ().

Collaborating

These libraries function within the voice assistant in a loop:

- The assistant uses speech recognition to listen for a command.
- The text that has been identified is processed.
- The assistant uses text to compose a response in response to the directive.
- Pyttsx3 plays the response text via the speakers after converting it to speech.

OS

Your voice assistant can interface with the underlying operating system to carry out different tasks in response to your voice commands by utilizing these OS module functionalities. Imagine invoking

the command “open photos” from your assistant, and it will utilize OS to start your favorite picture viewer program. It is imperative to have security. Because os.system runs arbitrary commands, use caution when working with it. To minimize security concerns, only accept trustworthy commands or inputs that have been confirmed by the user.

EEL

EEL fills the gap by letting you use Python's libraries and capabilities directly from within your graphical user interface. JavaScript can interface with Python functions to facilitate data interchange and communication.

Faster Development: EEL uses well-established web technologies that you may already be familiar with to speed the creation of GUIs.

Python Integration: You may easily incorporate the functionality and power of Python modules into your graphical user interface (GUI).

EEL applications have the ability to operate without an internet connection, in contrast to web applications. Along with vocal responses, EEL could be utilized for basic visual feedback systems. For example, responding to a request using synthesized speech and acknowledging it with an on-screen symbol.

Play Sound

Core Purpose: Playsound (sound) is the library's principal use. This method uses your system's native audio player to play an audio file (such as an mp3, wav, or other format) that you provide as input.

Play sound is capable of handling a number of audio formats that are frequently used with computers, such as MP3, WAV, and OGG.

Simpleness of Use: Adding the library to your Python programs is easy and it is lightweight. Its modest setup requirements make it perfect for novices or projects in which audio playback serves as a secondary purpose.

Playing Custom User-uploaded Audio: Play sound can manage playback if your assistant lets users submit their own audio files (such as wake words or greetings).

Simple Sound Effects (Limited Use): Play sound can be used to create simple sound effects that are loosely connected to speech output.

Play sound essentially offers rudimentary audio playing functionality, but other libraries offer more specific functions like text-to-speech and integrated audio cues for fundamental voice assistant uses.

PV PORCUPINE (version 1.9.5)

Pv Porcupine is a wake word engine made to run on smartphones with limited resources and be lightweight. It can be used to separate background noise from a particular keyword (such as “Alexa” or “Hey Google”). This is an essential initial step for voice-activated assistants since it enables the system to remain dormant and wait for the wake word, activating only upon hearing it.

Wake Word Detection

Your software will send out a signal or raise a flag if Pv Porcupine finds the wake word.

Voice Assistant Activation

Your application may turn on the primary voice assistant features after detecting a wake word. To detect and comprehend spoken commands from the user, this may include using a speech recognition engine instead of a typing engine.

RE

Regular expressions, or the RE library in Python, are essential for processing user queries in voice assistants. This is how it is typically applied:

Finding Keywords

Within the user's spoken command, you can use the RE command to look for particular keywords. To start the music playback feature, for instance, type "play music" into the search bar.

Details Extraction

Specific details can be taken out of a user's query using regular expressions. Assume someone asks, "What's the weather like in London today?" In this case, "London" might be used as the location and "today" as the duration of the weather request.

Verifying Formats

They can be used to determine whether or not the user's input adheres to a particular format. For example, you could make sure that when a user requests to "set a reminder for (date)", a date format such as "YYYYMMDD" is used.

HUG CHAT

Hug Chat is a Python library designed to simplify the process of creating APIs by minimizing boilerplate code and providing a straightforward syntax. It automates API documentation generation and leverages Python's type annotations for data serialization and deserialization, reducing manual parsing overhead. With Hug, developers can focus on application logic rather than infrastructure, thanks to its intuitive design and support for various input and output formats like JSON, HTML, and plain text. Additionally, Hug is highly extensible, allowing seamless integration with other Python libraries and frameworks. Whether building a basic RESTful API or a complex web service, Hug facilitates rapid development and improves overall productivity.

Pre-trained Models for Speech Recognition

Hugging Face provides Automatic Speech Recognition (ASR) models that have already been trained. These models can be used to translate user-spoken audio into text within your voice assistant. The user's intent can then be ascertained by further processing this text output.

PyWhatKit

PyWhatKit is a flexible Python library that works on different computerization errands, including WhatsApp informing, Google look, text-to-penmanship transformation, YouTube video playback, and from there, the sky is the limit. Its natural point of interaction and broad usefulness make it an important instrument for engineers hoping to smooth out their work processes and robotize dull errands proficiently. Whether you are fabricating a chat bot, robotizing online entertainment cooperations, or making mixed media applications, PyWhatKit gives the devices you want to take care of business successfully.

SQliteViewer

Visual Studio Code (VS Code), SQL Viewer is an extension that provides a convenient way to interact with SQL databases directly within the editor. It allows users to connect to various database management systems (DBMS) such as MySQL, PostgreSQL, SQLite, and more, and execute SQL queries directly from VS Code. SQL Viewer offers features like syntax highlighting for SQL queries, IntelliSense for auto-completion of SQL statements, and the ability to view query results in a tabular format. Users can also manage database connections, browse database schemas, and visualize data using charts and diagrams. With SQL Viewer, developers can streamline their workflow by seamlessly integrating database management tasks into their coding environment, enhancing productivity and efficiency.

PyAudio

PyAudio is a Python library that gives ties to PortAudio, a cross-stage library for sound info and result. It permits designers to effortlessly control sound streams, record sound from different information gadgets like amplifiers, and play back sound through speakers or earphones. PyAudio upholds an extensive variety of sound configurations and gives low-level admittance to sound gadget settings, empowering fine-grained command over sound handling boundaries. With PyAudio, designers can execute constant sound handling applications, construct voice acknowledgment frameworks, make sound impacts, and substantially more.

Its basic yet strong connection point, joined with its cross-stage similarity, pursues PyAudio a famous decision for sound related undertakings in Python improvement. Notwithstanding, it is actually important that PyAudio requires establishment of the PortAudio library, which might include some extra arrangement steps relying upon the stage (Tables 1–4, Figures 2–4).

Table 1. Test Case: 1.

S.N. of Test Case	Item
Name of Test	Checking Mic Button Functionality
Item/Feature being Tested	Mic Button
Sample Input	Tapping the Mic Button
Expected Output	Jarvis page should open
Actual Output	Jarvis page should open

Table 2. Test Case: 2.

S.N. of Test Case	2
Name of Test	Voice Recognition
Item/Feature being Tested	PyAudio
Sample Input	Tell me about you?
Expected Output	Spoken Sentence Display
Actual Output	Spoken Sentence Display
Remarks	Module is working properly

Table 3. Test Case: 3.

S.N. of Test Case	3
Name of Test	Chat Button Functionality
Item/Feature being Tested	Chat Button
Sample Input	Clicking on the chat button
Expected Output	Chat history should be shown
Actual Output	Chat history should be shown
Remarks	Module is working properly

Table 4. Test Case: 4.

S.N. of Test Case	4
Name of Test	HugChat Functionality
Item/Feature being Tested	Question Response
Sample Input	Who won the IPL match today?
Expected Output	Respected Answer is shown
Actual Output	Respected Answer is shown
Remarks	Module is working properly

Figure 2. Output or mic button functionality.

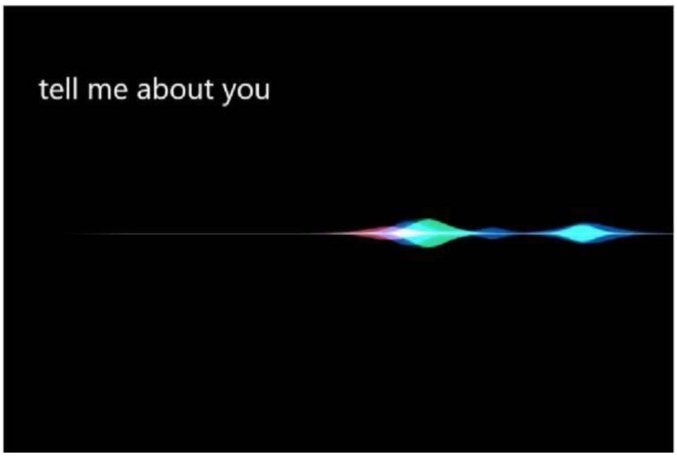

Figure 3. Output for voice recognition.

Figure 4. Output or hugchat functionality.

RESULT ANALYSIS

NLP-powered voice assistants are very advantageous for a wide range of uses (Tables 5–7, Figures 5–7). They serve as users' personal assistants, assisting with message sending, appointment scheduling, reminders, and information access [1, 2].

The partner is equipped for performing errands like sending messages, computerizing YouTube, gathering data from Wikipedia and Google, and other fundamental undertakings with simply a voice order [3, 4]. The study examines the modules and usefulness of the aide, including discourse acknowledgment, text-to-discourse transformation, setting extraction, and that is only the tip of the iceberg. The objective is to make a flexible and productive partner that can save time and help in an unexpected way abled people [5, 6].

Table 5. Test Case: 5.

S.N. of Test Case	5
Name of Test	Third Party WhatsApp messaging
Item/Feature being Tested	PyWhatKit
Sample Input	Hai Daddy!
Expected Output	Message is sent on WhatsApp
Actual Output	Message is sent on WhatsApp
Remarks	Module is working properly

Table 6. Test Case: 6.

S.N. of Test Case	6
Name of Test	Multilingual Functionality
Item/Feature being Tested	recognizing native language (Telugu)
Sample Input	Em chesthunav?
Expected Output	Respected Answer is shown
Actual Output	Respected Answer is shown
Remarks	Module is working properly

Table 7. Test Case: 7.

S.N. of Test Case	7
Name of Test	Multilingual Functionality
Item/Feature being Tested	recognizing native language (Kannada)
Sample Input	Oota ayetha?
Expected Output	Respected Answer is shown
Actual Output	Respected Answer is shown
Remarks	Module is working properly

(a)

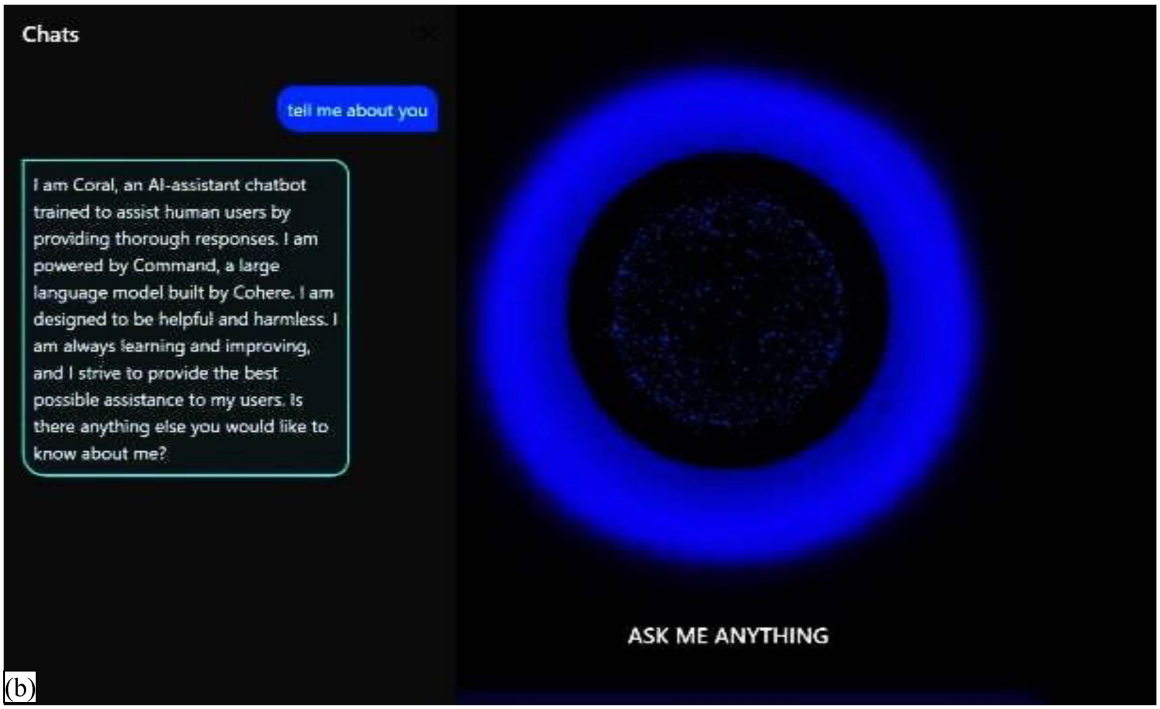

Figure 5. (a and b) Output for third party WhatsApp messaging.

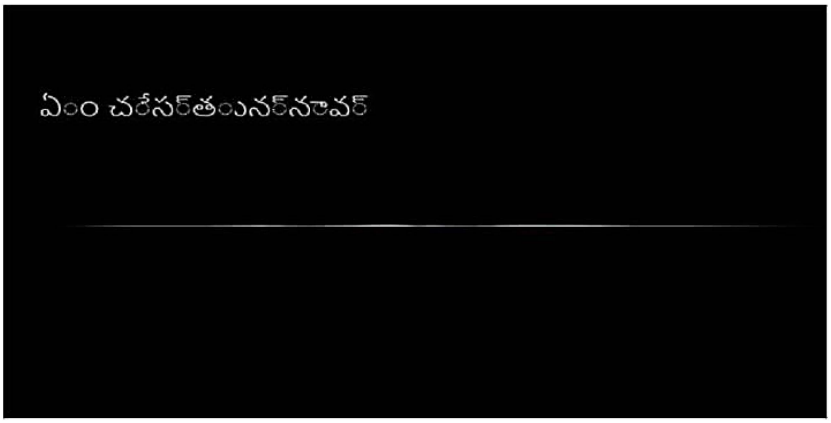

Figure 6. Output or Telugu language test.

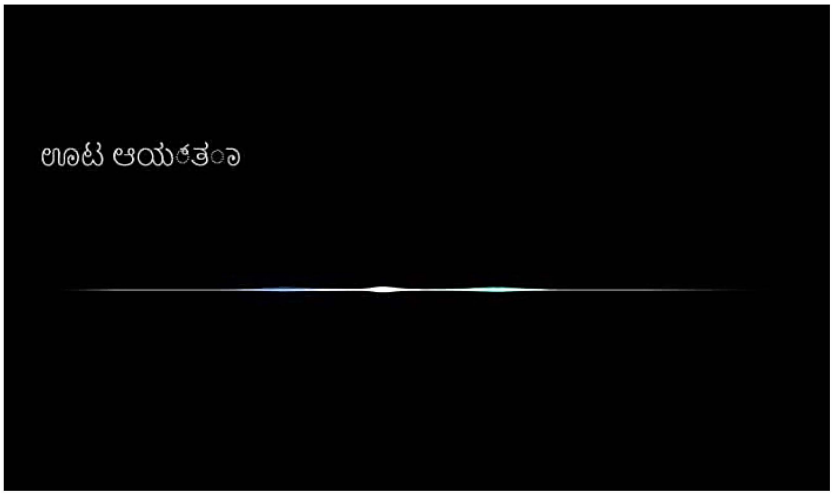

Figure 7. Output for Kannada language test.

CITATIONS

The study additionally makes reference to the effect of voice aides on client association with innovation and likely arrangements for further developing security and extending the partner to Android gadgets [7–9].

The study underscores the significance of wake words, discourse acknowledgment, and figuring out in the association among clients and Alexa. It additionally presents the construction of an Alexa Ability and the job of the Alexa Voice Administration in handling client demands. The exploration gives bits of knowledge into the exhibition and productivity of the discourse connection point and features the progressions in voice acknowledgment innovation [10–12].

With voice assistance, speech recognition is a game-changing technology that lets devices comprehend and interpret user commands or questions in spoken language. Its ability to provide a hands-free and natural way to communicate with technology completely changes the way people engage with it. To make this process easier, voice assistants like Cortana, Alexa, Google Assistant, and Siri use complex algorithms and machine learning models [13].

CONCLUSION

Using Python to Create a Flexible Voice Assistant: This study investigated Python's potential for creating a feature-rich, configurable voice assistant. We spoke about the essential features that a voice assistant must have and how Python libraries might help. We have successfully achieved connecting our voice assistant with our basic daily usage app like WhatsApp for calls and chats, moreover we have integrated it with a Chabot for collecting up to date information. We have also included a chat feature in the application so that it stores all the activities and saves it for our reference in future. We have achieved successfully recognizing multilingual languages, and in future, we would try to update it to a point where it can reply us in those multilingual languages.

Speech Recognition

Speech Recognition and similar libraries make it possible to translate spoken audio into text so that the assistant can comprehend user requests.

Natural Language Processing (NLP): Tools for applications like entity extraction, sentiment analysis, and intent categorization are available from robust libraries like NLTK and spaCy. These features aid in the assistant's interpretation of the intent behind the user's inquiries.

Text-to-Discourse (TTS)

Libraries like gTTS work with changing over the associate's reactions into regular sounding sound for client input.

Outside APIs and Structures: Python's huge environment permits mix with outer APIs and systems for functionalities like climate data, music playback, or shrewd home control, growing the associate's abilities.

Adaptability

Python's adaptability engages designers to fit the aide to explicit requirements and inclinations. Clients can characterize custom wake words, functionalities, and reaction styles.

We additionally featured the job of libraries like RE for ordinary articulation based order parsing and investigated how structures like PyAudio work with sound stream. Moreover, the capability of PvPorcupine for low-asset wake word identification and Embracing Face for pre-prepared models and NLP pipelines was examined.

Acknowledgement

We express our sincere thanks to our respected dean Dr. Md. Sameeruddin Khan, Dean, School of Computer Science Engineering and Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved professor Dr. L. Shakkeera, Associate Dean, School of Computer Science Engineering and Information Science, Dr. R Mahalakshmi, Professor and Head, School of Information Science, Presidency University for rendering timely help for the successful completion of this project.

We sincerely thank our project guide, Dr. J. Ghayathri, Associate Professor, School of SOIS, for her guidance, help and motivation. Apart from the area of work, we learnt a lot from her, which we are sure will be useful in different stages of our life. We would like to express our gratitude to Faculty Coordinators and Faculty, for their review and many helpful comments.

We would like to acknowledge the support and encouragement of our friends.

REFERENCES

1. Kumar A, Kaur D, Pathak AK. Voice assistant using python. In 2022 IEEE International Conference on Cyber Resilience (ICCR). 2022 Oct 6; 1–4.
2. N. Umapathi, G. Karthick, N. Venkateswaran, R. Jegadeesan, Dava Srinivas. Desktop's Virtual Assistant Using Python. Eur Chem Bull. 2023; 12(S3): 5975–5984.
3. Agarwal A, Mishra A, Tiwari S, Singh R. Virtual Voice Assistant using Python. Journal of Advanced Research in Information Technology, Systems and Management (JoARITSM). 2022 Nov 28; 6(1): 5–8.
4. Kumar A, Kaur D, Pathak AK. Voice assistant using python. In 2022 IEEE International Conference on Cyber Resilience (ICCR). 2022 Oct 6; 1–4.
5. Kulkarni V, Shreyas Kallurkar, Vipul Waikar, Patil S, Deshpande S, Kulkarni V, *et al.* Virtual Assistant Using Python. J Emerg Technol Innov Res (JETIR). 2014; 9(5): k198–201. Available from: https://www.jetir.org/view?paper=JETIR2205B23
6. Durge A, Lokhande A, Nagpure A, Dharmik C, Dhawale K. AI powered virtual voice assistant. Int Res J Mod Eng Technol Sci. 2023; 5(11): 796–800.
7. Huang Y. Research on the development of voice assistants in the era of artificial intelligence. In SHS Web of Conferences; EDP Sciences. 2023; 155: 03019.
8. Pramod Karande, Shubham Borchate, Bhavesh Chaudhary, Deveshree Wankhede. Virtual Desktop Assistant. Int J Res Appl Sci Eng Technol (IJRASET). 2022; 10(III): 1916–1920.
9. Sahu A, Jha A, Bhargava R, Priya P, Kumari R. Voice Assistant Using Artificial Intelligence. In Proceedings of the International Conference on Innovative Computing & Communication (ICICC). 2022 Mar 10.
10. Malodia S, Islam N, Kaur P, Dhir A. Why do people use Artificial Intelligence (AI)-enabled voice assistants?. IEEE Transactions on Engineering Management. 2021 Dec 6;71:491-505.
11. Kumar L, Yadav K, Singh J, Tripathi K. AI-based voice assistance using AWS. Int J Innov Res Comput Sci Technol (IJIRCST). 2021 May; 9(3): 77–82. ISSN. 2021:2347-5552.
12. Austerjost J, Porr M, Riedel N, Geier D, Becker T, Scheper T, Marquard D, Lindner P, Beutel S. Introducing a virtual assistant to the lab: A voice user interface for the intuitive control of laboratory instruments. SLAS Technol: Translating Life Sciences Innovation. 2018 Oct; 23(5): 476–482.
13. Preethi G, Abhishek K, Thiruppugal S, AVoice VD. Assistant using Artificial Intelligence. Int J Eng Res Technol (IJERT). 2022 May; 11(5): 453–457.

