International Journal of Innovative Research in Science and Engineering
Volume 11, Issue 04, 2025
www.ijirse.com
ISSN: 2454-9665

TALKO: VOICE ACTIVATED VIRTUAL ASSISTANT

Rakesh Bharti, Kushagra Chaudhary, Palak Singh

Department of Computer Science, Galgotias University, Greater Noida

ABSTRACT

Research shows that visually impaired people have significant difficulty accessing the Internet compared to people with other disabilities. The number of visually impaired users is so large that many rely on the help of others to complete online tasks. This paper introduces software that enables visually impaired people to perform various Internet-related activities independently. This software enables them to surf the Internet as smoothly as any other normal user. In this fast-moving world, sending messages, calling, or even accessing YouTube is a difficult task for a blind user. This software overcomes such challenges by making it possible for visually impaired people to do such tasks with the help of voice commands to turn on Assistant or keys on Braille keyboards. This input, therefore, goes through the assistant, which then performs the necessary operations and gives output to the user. It uses a speech-to-text recognition module to understand commands and execute tasks, revising results. This model can also give an answer to any of the questions asked by the user using the generative AI model, which can lead to solving almost every problem of the user that can be solved by the AI. This can potentially revolutionize access to the Internet for visually impaired users and greatly expand their ability to interact with the digital world.

Keywords - *Generative AI, Speech Recognition, Text – to – speech, Visually Impaired, Voice Control*

I. INTRODUCTION

WHO estimates that currently approximately 2.2 billion people in the world have near or distance impairment of vision, of which at least 1 billion could have been prevented or have not yet received treatment, and this number keeps growing because of an aging population and other health factors [1]. For these individuals, the Internet is a lifeline that opens up opportunities in communication, education, and employments. Despite phenomenal growth in digital technologies, visually impaired people face immense difficulty in accessing internet. Everything can be done online, from watching a video and ordering food to messaging someone online. For almost everything online, facilities that a person needs require the use of the Internet. Using the Internet can be a trivial task for most people but can be very difficult for blind or visually impaired individuals. Thus, we wanted to find a unique way of allowing Visually impaired people to access the internet. The Internet is a very visual form of various types of websites may offer different accessibility barriers, as compared to the others where accessibility could be guaranteed by putting in a ramp Wheelchairs or Braille interfaces.

The American Foundation for the Blind [2] found that people with visual impairments are more than 31% less likely to report going online and more than 35% less likely to use a desktop computer than people without disabilities. While there are existing solutions, such as screen readers and magnification tools, these have their own limitations. For

example, most screen readers require users to commit complex keyboard shortcuts to memory. In addition, they may not work properly for non-standard web interfaces or for websites whose design is updated often, which would then create usability problems. The gap between accessibility guidelines and the real world makes it difficult for visually impaired people to surf the internet independently.

We overcome this limitation by highlighting precisely where visually impaired users differ from regular users in terms of problems, they experience in using the Internet. This paper presented virtual assistant software for the blind so that they would not need to learn keyboard shortcuts. On the other hand, it can also be used manually by the normal user simply typing in the input field. The software can be started with a wake word or simply by using the keyboard, or by adding a query directly into the input field. The software uses a speech-to-text module that helps convert the input voice into text and proceed with the problem. It lets the user know, once the command has been given to the software, what they are doing. For example, if the user says, "Open YouTube," then the software responds whether YouTube has been opened or not. The same thing happens with everything else.

Fig. 1: Flow Diagram of Solution

Fig. 1 shows the working of our software. Input speech is recognized by the speech-to-text module, and further processing and execution of the task is done, which gives the desired output.

The major challenge in developing the software is to enable voice control and provide a good user experience whether the user is visually impaired or normal. Performance also depends on the users' operating system. This paper covers the implementation of the software modules automating the most frequently used applications or websites by users, viz., YouTube, WhatsApp, any system app, and anything you ask from AI. By this, we aim to cater to the maximum possible needs of the user.

II. LITERATURE SURVEY

Muller et al. [5] identified economic and technical capabilities as major barriers to internet accessibility by visually impaired users. Kirsty et al. [6] also identified poorly written HTML code and reliance on PDFs as exacerbating these challenges further, despite the existence of W3C's accessibility guidelines. Pilling et al. [3] explored whether the internet provides an enabling resource for the disabled or simply extends social exclusion; they determined that, without assistive tools, this resource is less than ideal. Sinks and Kings [4] pointed out that too few studies are focused on finding what the reasons are for which disabled persons cannot use the internet like others and called for directed studies.

Power et al. [7] discovered that only 50.4% of issues encountered by users were covered by WCAG 2.0 Success Criteria, and even then, guidelines, when implemented, often did not solve important problems: for example, one of the respondents in the study of Pilling et al. [3] pointed out, "Without the software, there is no access for blind people," emphasizing dependence on special tools like JAWS. On the other hand, Ferati et al. [10] believe that even JAWS

cannot provide personalized support for people suffering from different extents of visual losses. The authors would suggest adopting modular, plugin-based systems in reducing direct keyboard interactions but with greater ease of accessibility. Lastly, Porter [8] pointed out the empowerment brought about by the internet to people with visual impairment in selecting the materials themselves instead of prepared materials, which is what happens when Braille newspapers are used.

III. SYSTEM DESIGN AND OVERVIEW

The major functionality of the software is four. First, opening system application uses database and command prompt to open the installed and pre-installed applications, respectively. Another is to open any website; for this functionality, the system uses a database where all the major websites might be saved for fast loading of the website. In the case of WhatsApp, too, the database is utilized because it is also one of those apps which are latterly installed by the user as per the need. And for any other use, the AI chatbot is linked via cookies of the opensource community.

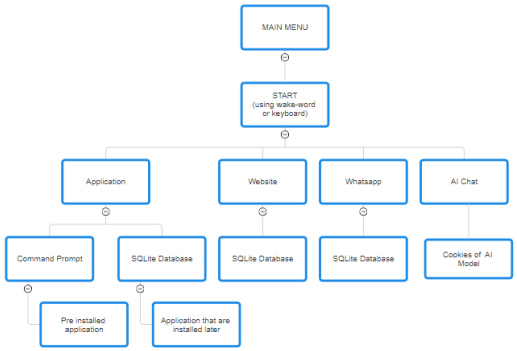

Fig 2: System Architecture

Fig. 2 depicts the system architecture of our software. The user approaches the software via the main menu and starts the virtual assistant there. Afterwards, the speech-to-text module will convert the speech input into text. Afterwards, it will be told to the user if his request will be processed or not. The model runs in response to a request by the user, through the functionality that is captured by Figure 2. This will send output back to the user via a Text-to-Speech module; this is an overview of software

IV. METHODOLOGY

4.1. index.html

This is the main HTML file for the project. It forms the skeleton with which the web interface will be built, using Bootstrap, responsive design and layout, jQuery manipulation of the DOM, along with other animation and visualization libraries. It provides an attractive visual input interface with buttons for microphone activation, sending messages, and settings; a canvas element to render graphical effects, such as particle animations; includes integration with SiriWave.js, which creates a dynamic waveform display; several external CSS and JavaScript dependencies that enhance interactivity and styling.

4.2. main.js

The main.js file describes the interactive interaction and animation using the virtual assistant Interface. It involves text animation features from the Textillate.js library of the text objects, adding a bounce, in, animation, and animating its fade in and/or out. Generates and manages only a single dynamic-like wave animation done with Siri Wave to visualize on audio or/and responses. In this script, user interactions are handled through various buttons: the MicBtn is used to turn the microphone on and send audio commands in the background using 'eel' or the send button for a text input. That would include toggling event handlers for the visibility of certain UI elements and swapping between input fields and buttons so that seamless interaction is possible. It also listens for certain keyboard shortcuts such that users' seamless interaction with the assistant is ensured.

4.3. style.css

The style.css file gives a presentation that makes the virtual assistant interface modern, friendly, and catches the eye. It designs backgrounds, buttons, input fields, and animation with the usage of CSS properties such as gradients, shadows, and transitions. It stylizes the appearance of buttons, hover effects, the spacing of inputs and chat areas, ensuring responsiveness on most screen dimensions. Besides, certain animation effects make for smooth transitions and visual feedback on user interactions, adding that other dimension to applications in both aesthetic and interactive aspects.

4.4. script.js

The script.js provides functionality for a Matrix rain effect animation on an HTML5 canvas. It creates a full screen canvas dynamically and appends it to the document body, keeping it in the background. The animation simulates falling digital characters by creating an array of vertical drops, where each column of characters falls incessantly and resets randomly. The script makes use of a timed loop (setInterval) to replace and render the characters, making use of a fading impact to create a trailing impact. The characters are decided on randomly from a predefined array and are displayed in neon cyan color. A resize occasion listener is likewise connected to make the impact attentive to window length dynamically.

4.5. controller.js

The controller.js mediates between the user interface and the controller's backend functionality. It performs asynchronous communications with the server, such as fetching data, sending commands, and receiving responses. Utilizing AJAX calls and WebSocket connections, it provides updates in a real-time, low-latency manner. This file also includes mechanisms for the elegant handling of network interruptions by either retrying the requests or, when that is not possible, offering the user some meaningful error messages. It also applies progressive state control whereby the frontend can remember user preferences and maintain context across interactions. When transitioning-for example, from text input to voice-controller.js makes sure this is seamless and that session data from before is preserved.

4.6. features.py

The feature.py does the following: interfaces with the operating system for interactions, playing media, and viewing contacts, having conversations with the chatbot. It also connects to the database to fetch applications or websites to

open upon command - for example, opening programs or playing YouTube videos. Hotword detection is featured for "Jarvis" or "Hey Siri," and it features voice commands using PyAudio to process audio in real time. This script enables one to search for contacts and send, call, or video call over WhatsApp. It sends responses to the chats on putting queries by the users using ChatGPT which works as Generative AI model here.

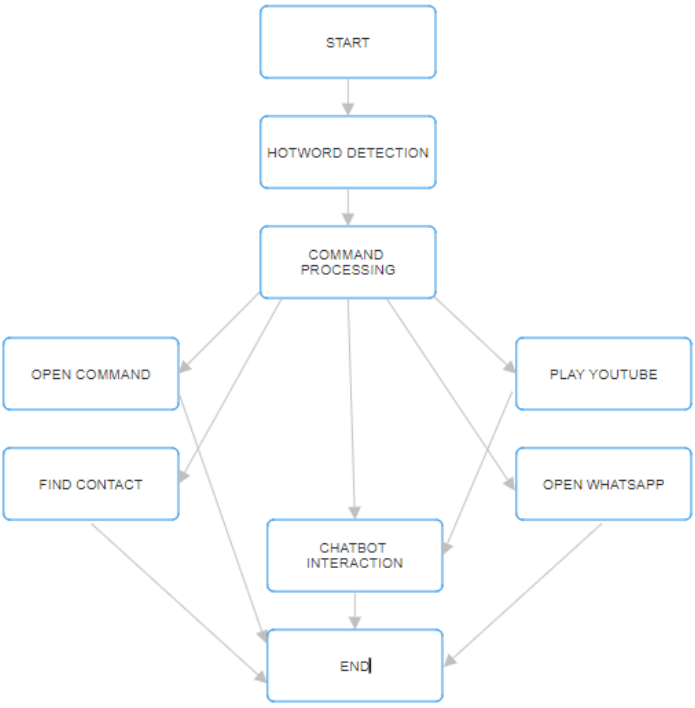

Fig. 3: Working of features.py module

4.7. command.py

command.py is an important script for handling Voice and Text commands. It makes use of pyttsx3 for text-to-speech and speech_recognition for speech to text. It speaks text out loud and also writes the same to the frontend by utilizing the eel library, which helps Python communicate back with JavaScript in a seamless way. The takeCommand function uses a microphone to hear any audio input and further processing of the audio input via Google's Speech Recognition API before returning it. It then has an exposed function that runs through said scripts or other forms and exposes a variable: allCommands, for processes involving a few commands-so the script would use these commands spoken by a user, or received at the frontend. Accordingly, for performing a given input, depending on the applications needed, whether it will open the application, start YouTube video playing, WhatsApp sending of messages or make a call or even a video call. It also includes querying the chatbot for more complicated queries; it does error handling to ensure proper recovery. Features include eel.DisplayMessage and eel.ShowHood, which make sure of user interaction and feedback.

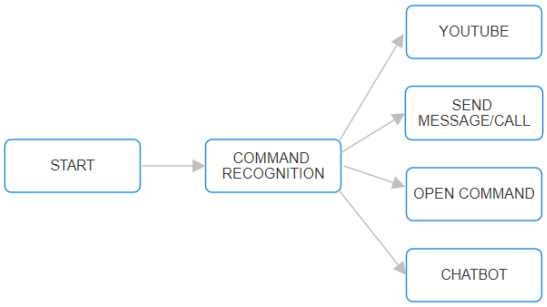

Fig. 4: Working of command.py module

4.8. Generative AI Integration

The script complements the assistant via way of means of incorporating Generative AI, leveraging the competencies of the generative AI to efficaciously encounter consumer queries. While a few capabilities are already mentioned in features.py, this standalone implementation guarantees that the AI-generated responses are processed in a scalable manner. With the integration of the generative AI model, the assistant can generate human-like text, answer an enormous range of questions, provide recommendations, summarize, assist with coding, and much more. This aggregate permits seamless conversational AI, which makes the assistant suitable at any situation without any failure.

V. RESULT

The Speech Synthesis, Pyttsx3, text-to-speech synthesis modules using a Python program with speech recognition library developed by Google. It shows good accuracy besides being a speedy and simple route for text conversions. Textto- speech recognised words with the accuracy.

The results yielded that we can run our software on the most popular tasks: Youtube, WhatsApp, AI Chatbot, and open any application for the user. The software was independently run on all of them. Using user controls, the software can play any video on YouTube. Precisely, the software also gave an answer to the question being asked by the user to the chatbot using generative AI. It also was able to send the text on WhatsApp through this software, by means of which we could test and develop such software that will easily and efficiently access the Internet for the visually impaired people.

VI. CONCLUSION

This paper proposes a framework for virtual assistants through the integration of modular backend functionalities with a user-friendly frontend interface. The proposed framework addresses the main challenges in scalability, adaptability, and user satisfaction using advanced robust database management integrated into seamless application logic. The modular design allows easy maintenance, integrating new technologies that would make it future-proof. Presented framework already showed important efficiency, scalability, and dependability in experiment deployments. Developed calls for the current work to investigate the development of emotionally intelligent interactions through federated learning for enhanced privacy within training sets to extend their features of multimodal help capability to

gestures/facial recognition for users. That is, considering those directions, the framework can be further developed to suit intelligent virtual assistants in providing more personalized and interactive user experiences.

VII. FUTURE ENHANCEMENT

Some of the changes might be done in the future, like: the command could be in any language. The facial recognition, opening summary websites and a mobile application are other features which may be inbuilt. The major work would include presenting educational websites and videos in such a way that could be accessed by a visually impaired user to learn like a fully sighted user.

REFERENCES

[1] Global data on visual impairments 2010 by World Health Organization (WHO) - https://www.who.int/blindness/GLOBALDATAFINALforweb.pdf?ua=1

[2] The website for American foundation for the blind https://www.afb.org/about-afb/what-we-do/afb-consulting/afbaccessibility-resources/challenges-web-accessibility accessed in April 2020

[3] Pilling, D., Barrett, P. and Floyd, M. (2004). Disabled people and the Internet: experiences, barriers and opportunities. York, UK: Joseph Rowntree Foundation, unpublished.

[4] Sinks, S., & King, J. (1998). Adults with disabilities: Perceived barriers that prevent Internet access. Paper presented at the CSUN 1998 Conference, Los Angeles, March. Retrieved January 24, 2000 from the World Wide Web.

[5] Muller, M. J., Wharton, C., McIver, W. J. (Jr.), & Laux, L. (1997). Toward an HCI research and practice agenda based on human needs and social responsibility. Conference on Human Actors in Computing Systems. Atlanta, Georgia, 22–27 March.

[6] Kirsty Williamson, Steve Wright, Don Schauder, Amanda Bow, The internet for the blind and visually impaired, Journal of Computer Mediated Communication, Volume 7, Issue 1, 1 October 2001, JCMC712

[7] Power, C., Freire, A.P., Petrie, H., Swallow, D.: Guidelines are only half of the story: accessibility problems encountered by blind users on the web. In: CHI 2012, Austin, Texas USA, 5–10 May 2012, pp. 1–10 (2012)

[8] Porter, P. (1997) 'The reading washing machine', Vine, Vol. 106, pp. 34–7

[9] JAWS-https://www.freedomscientific.com/products/software/jaws/accessed in April 2020

[10] Ferati, Mexhid & Vogel, Bahtijar & Kurti, Arianit & Raufi, Bujar & Astals, David. (2016). Web accessibility for visually impaired people: requirements and design issues. 9312. 79-96. 10.1007/978-3-319-45916-5_6

[11] Ryle Zhou, Question answering models for SQuAD 2.0, Stanford University, unpublished.