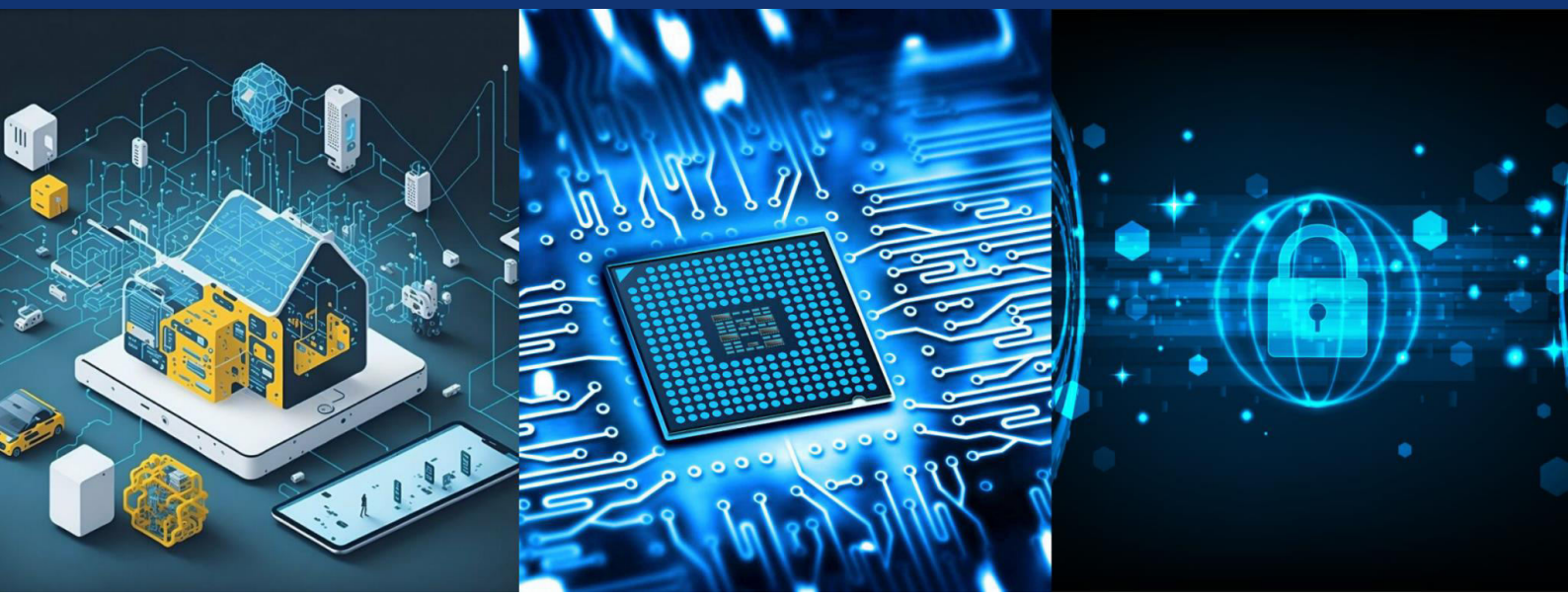
IJIRCCE

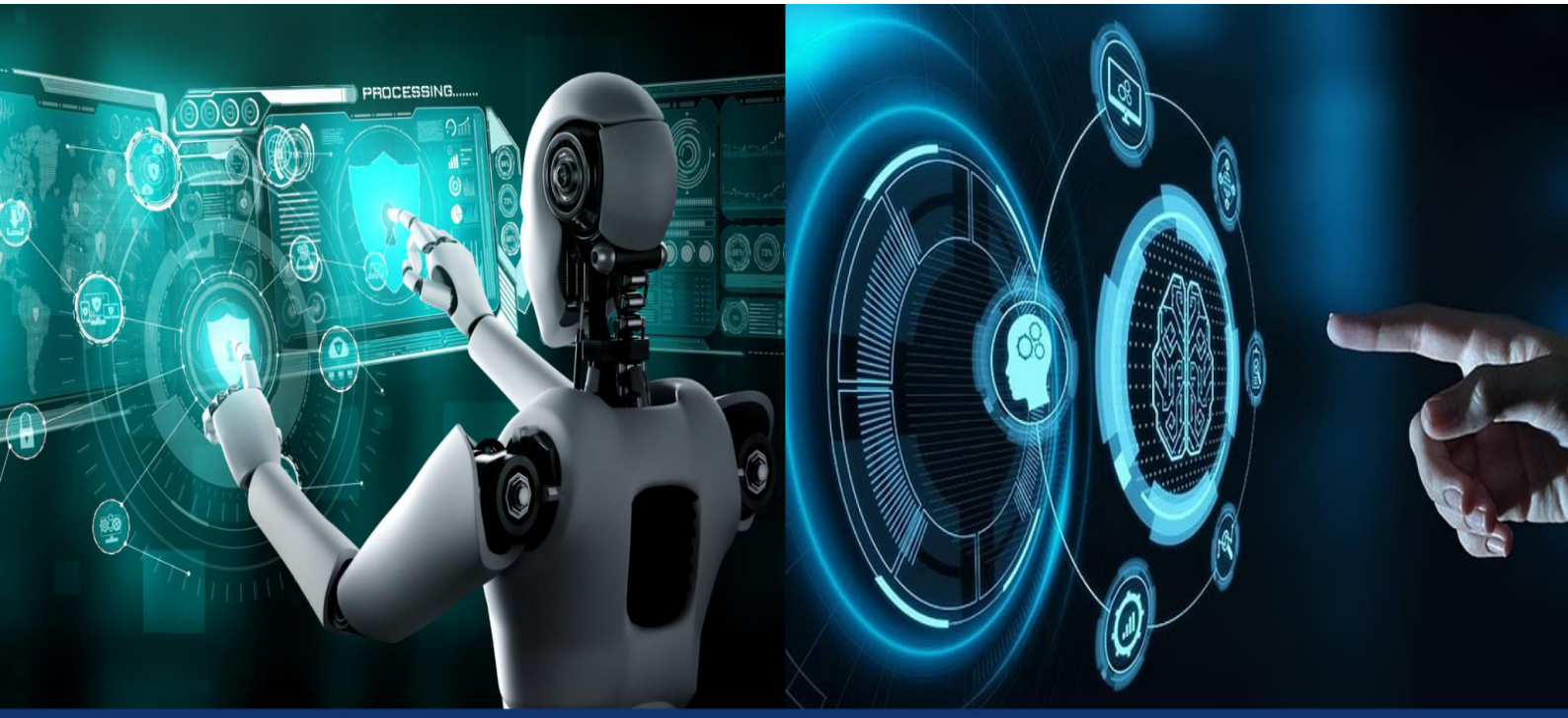
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771

Volume 13, Issue 5, May 2025

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305036

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Voice Controlled Personal Assistant

Akshatha A, Bhuvana Gowda N U, Dr. N.G. Goudru, Sharanya D

Department of Information Science and Engineering, Sambhram Institute of Technology, Bengaluru, India

ABSTRACT: In recent years, voice-controlled systems have gained significant attention due to their intuitive interaction and growing integration in smart devices. This paper presents the design and development of a voice-controlled personal assistant that leverages AI-based chatbot functionality to perform user-defined tasks through natural language commands. The system incorporates speech recognition, natural language processing (NLP), and text-to-speech components to ensure seamless two-way interaction between users and machines. Implemented using open-source tools and frameworks, the assistant demonstrates capabilities such as web search, time and date reporting, and simple task automation. The research aims to enhance user experience with minimal manual input, ultimately contributing to the advancement of intelligent human-computer interaction systems.

KEYWORDS: Voice Assistant, AI Chatbot, NLP, Speech Recognition, Human-Computer Interaction, Text-to-Speech, Personal Assistant

I. INTRODUCTION

In the age of rapid digital transformation, the way humans interact with machines has evolved significantly. One of the most promising advancements in this domain is the development of voice-controlled personal assistants. These systems allow users to perform various tasks through spoken commands, eliminating the need for traditional input devices such as keyboards or touchscreens.

Voice assistants have become increasingly popular due to their convenience and efficiency. Major technology companies have already deployed products such as Google Assistant, Amazon Alexa, and Apple Siri, setting a benchmark for intelligent systems capable of natural human interaction. Inspired by these developments, this paper presents the design and implementation of a custom-built voice-controlled personal assistant using artificial intelligence (AI)-based chatbot architecture.

The core components of the system include speech recognition, natural language processing (NLP), and text-to-speech (TTS) modules. Developed using Python and integrated with various open-source APIs, the assistant is capable of understanding voice commands, processing them using NLP techniques, and providing meaningful responses. The assistant is also equipped to perform actions such as retrieving information from the web, reporting the time and date, and executing basic automation tasks.

The objective of this project is to provide an affordable, lightweight, and customizable alternative to commercial voice assistants. By leveraging widely available programming libraries and APIs, the system demonstrates how voice interaction and AI chatbots can be integrated to create a functional and intelligent personal assistant.

II. LITERATURE REVIEW

Voice-controlled personal assistants have gained widespread adoption due to their ability to interpret and respond to human commands using natural language. Several commercial solutions like Google Assistant, Amazon Alexa, Apple Siri, and Microsoft Cortana have set industry standards for smart virtual assistants. These systems utilize advanced machine learning models, cloud computing, and proprietary data to ensure highly accurate and context-aware interactions.

In academic research, numerous studies have explored the development of voice assistants using open-source tools. For

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

instance, works such as [1] and [2] describe the integration of speech recognition and NLP models using Python-based libraries like Speech Recognition, NLTK, and spa Cy. These studies emphasize the feasibility of building voice interfaces without depending on commercial ecosystems.

Other research has focused on natural language understanding (NLU), where chatbot systems are trained to interpret user intent and generate appropriate responses. Dialog flow and Rasa have been popular frameworks in this domain, offering conversational flow control and intent recognition [3].

Despite these advancements, many existing solutions are either dependent on cloud services or are limited in terms of customization and offline functionality. Moreover, there is a gap in systems that provide lightweight, local processing voice assistants for educational or personal use. This project addresses this gap by developing a Python-based personal assistant that uses open APIs and NLP techniques to execute basic tasks through speech commands in a user-friendly manner.

III. SYSTEM DESIGN AND ARCHITECTURE

The architecture of the voice-controlled personal assistant is designed to follow a linear yet interactive flow that begins with the user's speech input and ends with either a spoken response or the execution of a task. The system is divided into five major components: User Interaction, Speech Recognition, Natural Language Processing, Command Execution, and Text-to-Speech Output.

3.1. User Interaction

The process starts with the user speaking into the system's microphone. This interaction acts as the primary input source and initiates the assistant's workflow.

3.2. Speech Recognition

The audio input is captured and converted into text using Python's Speech Recognition library, integrated with APIs such as Google Speech-to-Text. This step allows the system to interpret what the user has said in a readable and processable format.

3.3. Natural Language Processing

The recognized text is then passed to the NLP module, where Python-based libraries like NLTK and spa Cy help analyse and interpret user intent. Commands are parsed using rule-based logic to identify the task to be performed.

3.4. Command Execution

Once the intent is identified, the system executes the appropriate action. These actions include:

- Fetching the current date or time
- Performing a Google search
- Opening specific websites or applications
- Responding with text-based answers

Python's built-in operating system and web browser libraries are used for system and web-level operations.

3.5. Text-to-Speech Output

Finally, the system delivers a response using Python's **pyttsx3** library for offline text-to-speech conversion. This ensures users receive verbal feedback or confirmation, completing the interaction loop.

3.6. Architecture Diagram

The following diagram illustrates the flow of information and control within the assistant:

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305036

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

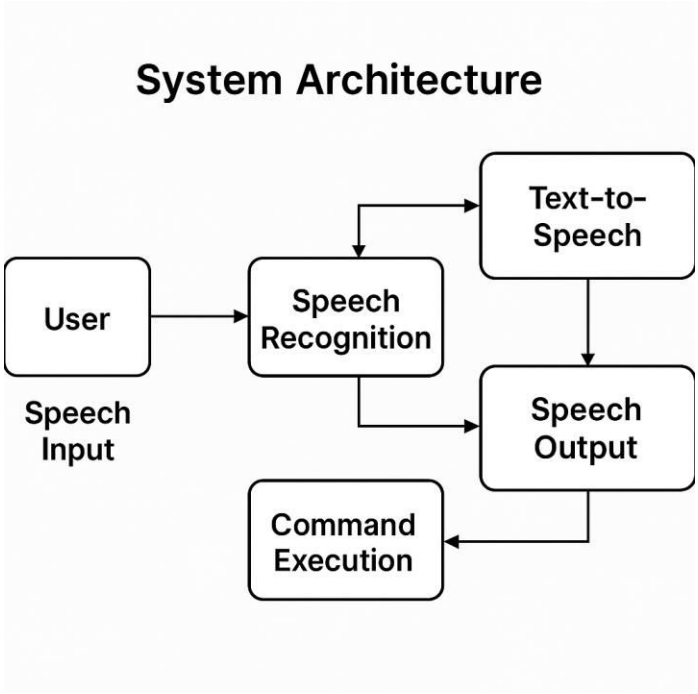


Fig.1 The proposed system's workflow diagram.

IV. METHODOLOGY

The development of the voice-controlled personal assistant followed a modular and iterative approach, focusing on integrating speech input, natural language understanding, response generation, and speech output. The system was built using Python as the primary programming language due to its simplicity, extensive library support, and wide adoption in the AI and NLP communities.
The methodology can be divided into the following key stages:

4.1. Speech Recognition

The first step involves capturing and interpreting user speech input. The Speech Recognition library in Python was used for converting audio input to text. The library integrates well with APIs like Google Speech-to-Text, which provides accurate transcription services for spoken language.

4.2. Natural Language Processing (NLP)

Once the speech is converted to text, it is passed to the NLP engine to understand the user's intent. Python's Natural Language Toolkit (NLTK) and spa Cy were employed to perform basic text preprocessing, such as tokenization, stop-word removal, and named entity recognition (NER). A rule-based approach was used to match commands to predefined actions (e.g., "What is the time?", "Open Google").

4.3. Command Execution

Based on the interpreted intent, the assistant executes corresponding actions. These include:

- Fetching the current time and date
- Performing a web search
- Opening websites or applications
- Responding to greetings or simple queries

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305036

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

APIs and system-level commands were used for executing tasks. For example, web browser module in Python was used to open URLs, while the operating system module helped trigger local applications.

4.4. Text-to-Speech (TTS)

To respond to the user audibly, the assistant converts textual responses to speech using the pyttsx3 library, an offline TTS engine compatible with multiple platforms.

4.5. Integration and Control Flow

All the modules were integrated into a single control loop, where the system continuously listens for voice input, processes the text, performs an action, and responds using speech output. Exception handling was incorporated to manage invalid commands or connectivity issues.

V. IMPLEMENTATION

The implementation of the voice-controlled personal assistant involved a combination of Python libraries, APIs, and AI techniques. The goal was to create a system that is functional, responsive, and easy to understand and customize for further development. The assistant was developed and tested on a local machine running a standard Python environment.

5.1. Programming Language:

- **Python 3.x**: Chosen for its simplicity, readability, and extensive library support for AI and NLP applications.

5.2. Libraries and Frameworks

- **Speech Recognition**: Used for capturing and converting voice input into text. It supports various backends, and the implementation used Google Speech Recognition API for accurate results.
- **NLTK (Natural Language Toolkit)** and spa Cy: These NLP libraries were used for preprocessing user input, such as tokenization and keyword extraction, to identify commands.
- **pyttsx3**: A text-to-speech conversion library that works offline and supports multiple speech engines. It provided vocal responses to the user's input.
- **OS**: This module allowed the assistant to interact with the operating system for executing local commands like opening applications.
- **Web browser**: Used to perform search operations or open websites in the default browser based on user commands.

5.3. APIs Used

- **Google Speech-to-Text API**: Provided robust and accurate voice-to-text conversion using online services.
- **Web Search APIs**: For extended functionalities, such as fetching Wikipedia results or weather information, APIs like wiki pedia or requests can be integrated.

5.4. Development Environment

- **Platform**: Windows/Linux OS
- **Tools**: Python IDEs VS Code was used during development.
- **Microphone**: Standard laptop/PC microphone or headset was used to receive voice input.

5.5. Custom Command Handling

A simple rule-based approach was implemented to map user queries to specific actions. The command handler checks for keywords (e.g., "time", "date", "search", "open") and routes the task accordingly.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305036

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

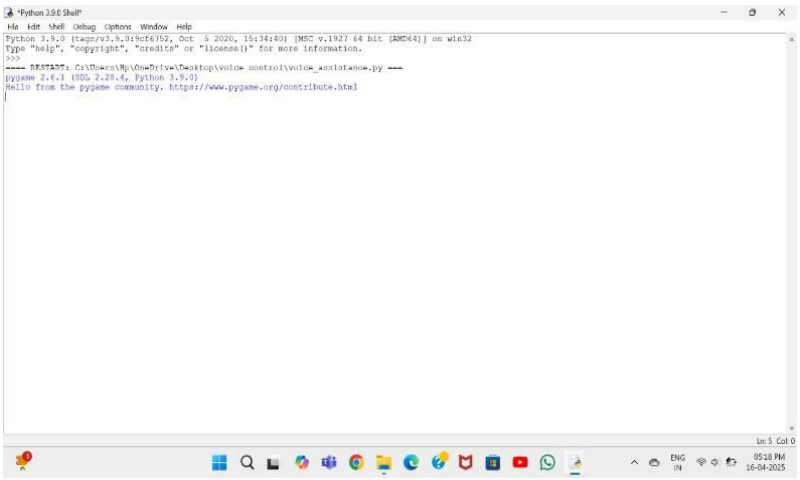


Fig.2. Screen of Interface

1. **Setting Alarms:** The assistant can set alarms based on user input. We can use the time and datetime libraries to set alarms and notify the user at the specified time.
2. **Control System Applications:** Using the pyautogui library, the assistant can open system applications such as Notepad, WordPad, or Excel.
3. **Weather Forecast:** To get the weather forecast, we can integrate a weather API like Open Weather Map. First, sign up for an API key on Open Weather Map.
4. **Date and Time**: To retrieve the current date and time, the datetime library can be used.
5. **Opening YouTube Videos**: The assistant can open YouTube videos using the web browser library. The web browser function allows the assistant to launch YouTube with a specific video.

Fig.3. Intent command page.

1. **Fetching Data from Google and Wikipedia**: To fetch data from Google or Wikipedia, we can use web scraping or APIs. For simplicity, we'll use Google search results and Wikipedia's API.
2. **Getting News:** Using an API To fetch the latest news, you can use APIs like News API.
3. **Sending WhatsApp Messages:** To send WhatsApp messages, we can use the pywhatkit library.
4. **Sending Gmail:** To send emails via Gmail, we can use Python's smtplib library.
5. **Main Program**: The main program will integrate all the components and loop continuously to listen for voice commands. Based on the recognized command, it will invoke the corresponding function.

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305036

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

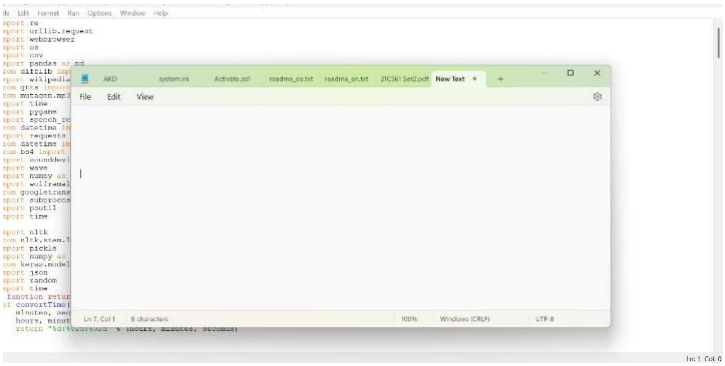


Fig.4. Opening notepad from command

VI. DISCUSSION AND LIMITATIONS

The voice-controlled personal assistant was successfully developed and tested in a controlled environment using a personal computer with Python installed. The assistant responded accurately to various voice commands, demonstrating reliable speech recognition, natural language understanding, and appropriate task execution.

A. Functional Results

The system was tested using a predefined set of commands across multiple categories. Key features and outcomes include:

Table 1. System Features

Command Type	Example Command	System Response
Time and Date Retrieval	What time is it?	"The current time is 12:00 PM"
Web Search	Search for Wikipedia	"Open default browser with search query"
Open website	Open YouTube	"Launches YouTube in browser"
Greetings and general query	Hello how are you?	"I'm doing great. How can I help you?"

© 2025 IJIRCCE | Volume 13, Issue 5, May 2025| DOI:10.15680/IJIRCCE.2025.1305036

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Application Launch	Open calculator	Launches calculator application

The assistant exhibited a response time of approximately 1–2 seconds, depending on internet connectivity and command complexity.

B. Accuracy

- **Speech Recognition Accuracy**: ~90% under clear voice and quiet background conditions.
- **Command Recognition Accuracy**: ~85% with rule-based keyword matching.
- **TTS Output Quality**: Clear and natural using the pyttsx3 engine.

C. Limitations

- The system may not perform optimally in noisy environments.
- It relies on rule-based NLP, which limits the understanding of complex or ambiguous commands.
- The assistant requires internet connectivity for speech recognition through the Google API.
- No GUI interface was implemented; interaction is entirely voice/text-based via the console.

D. Observations

- Performance can be improved using more advanced NLP techniques such as intent classification via machine learning models.
- The use of local/offline speech-to-text models could enhance privacy and reduce reliance on cloud services.
- Scalability is feasible with the integration of more APIs and modular command handling.

VII. CONCLUSION AND FUTURE WORK

This paper presented the design and development of a voice-controlled personal assistant based on artificial intelligence and natural language processing. The system was successfully implemented using Python, integrating key modules such as speech recognition, NLP, command execution, and text-to-speech synthesis. It enabled users to interact with their computer systems through simple spoken commands, achieving a seamless voice-based interface for basic task automation.

The assistant performed reliably across various command categories, demonstrating satisfactory accuracy in speech-to-text conversion and command recognition. By utilizing open-source libraries and APIs, the project remained accessible, lightweight, and customizable, making it an ideal starting point for educational and personal use.

However, the current implementation has certain limitations, such as dependence on internet connectivity for speech recognition and limited flexibility in understanding complex user intents. These areas present opportunities for future work.

Future Enhancements:

- **Incorporation of Machine Learning Models**: Implementing intent classification and contextual understanding using models like BERT or Rasa NLU can improve conversational abilities.
- **Offline Speech Recognition**: Integrating tools such as Vosk or CMU Sphinx could help develop a fully offline version for improved privacy and portability.
- **GUI Integration**: A simple graphical interface could make the assistant more user-friendly and visually appealing.
- **Task Expansion**: Adding support for reminders, file handling, calendar integration, and smart device control could expand the assistant's utility.

In conclusion, the project lays a foundational framework for building intelligent voice-controlled systems and can serve as a base for more complex and scalable AI-driven assistant technologies.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

REFERENCES

1. Smith, A., & Kumar, R. (2019). Design of a Voice Recognition-Based Virtual Assistant Using Python. International Journal of Computer Applications, 182(35), 25–30.
2. Patel, S., & Mehta, D. (2020). Natural Language Processing for Chatbot Development: A Comparative Study. International Journal of Artificial Intelligence and Applications, 11(2), 15–21.
3. Gupta, A. (2021). Conversational AI: An Overview of Tools and Techniques in NLP Chatbots. Journal of Emerging Technologies and Innovative Research, 8(5), 45–50.
4. Vaidya, R., & Sharma, P. (2022). A Review on Voice Assistants: Capabilities, Limitations, and Future Scope. International Journal of Scientific Research in Engineering and Management, 6(3), 78–84
5. AccuWeather: Accuweather apis. https://developer.accuweather.com/apis (2021)
6. Bocklisch, T., Faulkner, J., Pawlowski, N., Nichol, A.: Rasa: Open source language understanding and dialogue management. arXiv preprint arXiv:1712.05181 (2017)
7. Chowdhury, S.S., Talukdar, A., Mahmud, A., Rahman, T.: Domain specific intelligent personal assistant with bilingual voice command processing. In: IEEE Region 10 Conference (TENCON). pp. 731–734 (2018)
8. Coyne L., Gopalakrishnan S., S.J.R.I.: Ibm private, public, and hybrid cloud storage solutions. http://www.redbooks.ibm.com/redpapers/pdfs/redp4873.pdf (2014)
9. Facebook: React native. https://github.com/facebook/react-native (2021)
10. Felix, S.M., Kumar, S., Veeramuthu, A.: A smart personal ai assistant for visually impaired people. In: 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI). pp. 1245–1250 (2018)
11. Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al.: Deep speech: Scaling up end-toend speech recognition. arXiv preprint arXiv:1412.5567 (2014)
12. Honnibal, M., Montani, I.: spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing (2017)
13. Iannizzotto, G., Bello, L.L., Nucita, A., Grasso, G.M.: A vision and speech enabled, customizable, virtual assistant for smart environments. In: 2018 11th International Conference on Human System Interaction (HSI). pp. 50–56. IEEE (2018)
14. Khattar, S., Sachdeva, A., Kumar, R., Gupta, R.: Smart home with virtual assistant using raspberry pi. In: 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). pp. 576–579. IEEE (2019)
15. Matsuyama, Y., Bhardwaj, A., Zhao, R., Romeo, O., Akoju, S., Cassell, J.: Sociallyaware animated intelligent personal assistant agent. In: Proceedings of the 17th annual meeting.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details