RESEARCH PAPER

Voice Assistant Systems Developed Using Python: Architecture, Technologies, Challenges, and Future Directions

Prakhar Rai[1], Priyanshu[2], Vansh Palliwal[3], Yash Kumar[4], Dr. Punit Mittal[5]

[1,2,3,4,5] Department of Computer Science and Information Technology, Meerut Institute of Engineering & Technology, Meerut, U.P., India

Emails: prakhar.rai.csit.2022@miet.ac.in, priyanshu.jaikaran.csit.2022@miet.ac.in, vansh.palliwal.csit.2022@miet.ac.in, yash.kumar.csit.2022@miet.ac.in

Dr. Punit Mittal - Email: punit.mittal@miet.ac.in

Received: 17th Mar, 2026 | Revised: 29th Mar, 2026 | Accepted: 19th Apr, 2026 | Available Online: 5th May, 2026

ABSTRACT

The voice assistant systems have become an integral part of modern human–computer interaction, supporting communication between users and digital systems based on seamless languages. But besides some higher-level programming languages such as Python have become the most widely used in building systems of this nature with simplicity and covering architecture, underlying technologies, development systems and ecosystem, challenges and measurement that evaluates speech recognition accuracy, response delay, task completion success rate and user satisfaction along with large libraries support, also becoming natural to working tool kits like AI and machine learning frameworks that power these systems. We have conducted an extensive survey of existing Python-based voice assistant systems. Experimental results demonstrate the effectiveness of Python as a programming language for academic research in this domain.

Keywords: Voice assistant; Python; Speech recognition; Natural language processing; Artificial intelligence.

Index Terms: Voice Assistant; Python; Speech Recognition; Completion success rate; User satisfaction.

How to cite this article: Rai P, Priyanshu, Palliwal V, Kumar Y, Mittal P. Voice Assistant Systems Developed Using Python: Architecture, Technologies, Challenges, and Future Directions. Int J Drug Deliv Technol. 2026;16(42s): 806-816. DOI: 10.25258/ijddt.16.42s.90

Source of support: Nil.

Conflict of interest: None

1. Introduction

Human–computer interaction has evolved from command-line to graphical user interfaces, and more recently to natural language based interaction systems. Voice assistants are a great step forward in that development and allow humans to communicate with computers through speech rather than by entering data on a keyboard or touchscreen, as has always been done before." This shift in paradigm has increased the accessibility, alternative to these elder citizens, visually impaired(low vision)book readers and even not-so-tech-savvy folks.

The rapid adoption of voice assistant technologies around the world are directly as- sociated with the fast-growing of artifi- cial intelligence, cloud computing and ubiquitous smart devices. Commercial systems such as Google Assistant, Amazon Alexa and Apple Siri demonstrate the wide range of voice-related interactions. The clean readability and a very fast development cycle have also made Python an attractive

A. Motivation and Problem Statement

Commercial voice assistants are on there rise but there still a lot of technical architecture and implementation details left unfilled for systems built on Python. Algorithms are usually proprietary and even fully experimental evaluations in academic research are scarce. This Paper Addresses These Gaps With:

• Architectural evaluation of python- corresponding voice assistant systems

• Analysis of the currently available Python libraries and frameworks.

• Experimental prototype quantitative performance metrics

• Perspectives on key issues and future research!

B. Research Contributions

We highlight the major contributions of this

1) General architecture of a typical Python-based

voice assistant systems

2) Performance analysis through design and implementa- tion of Experimental Architecture

3) Quantitative results with an analysis of accuracy, latency, task success rate and user satisfaction

4) annotation of key obstacles and prospects for further inventiveness in voice assist- ant evolution

C. Paper Organization

The remainder of this paper is organized as follows: Section II presents background and related work. On the other hand, for example, hardware and software make advanced design of Pneuma that can complete them, it is necessary to have both hard- ersistence be realized by a Python-based voice assistant system. Section V outlines the experimental setup and VI gives Result analysis. Section VII addresses challenges whereas Section VIII describes future directions. Section XII concludes the paper.

II. Background and Related Work

Early models of spoken input date back to the 1960s, including IBM Shoebox which could recognise around a dozen spoken digits. Subsequent work employed statistical models such as Hidden Markov Models (HMMs) in an effort to improve the generation of text characters. Next came deep learn- ing (a watershed moment) that enabled the systems to discover new features of speech from huge datasets.

Historical Development of Voice Recognition

Speech recognition technol- ogy developed in several stages, and these can be summarized as:

950s-1960s (Early Systems) : Bell Labs' Audrey system (1952) could recognize digits spoken by a single speaker. IBM Shoebox (1962) understood 16 words.

1970s-1980s (Statistical Methods): The introduc- tion of Hidden Markov Models (HMMs) by Baker (1975) and Jelinek (1976) revolutionized the field.

1990s–2000s (Commercial Systems): Dragon NaturallySpeaking (1997) brought continuous speech recognition to consumers.

2010s-Present (Deep Learning Era): In- troduction of neural networks, LSTM, and transformer models achieved near-human ac- curacy.

Modern Voice Assistant Platforms

Table I compares major commercial voice assis- tant platforms.

TABLE I: Comparison of Major Voice Assistant Platforms

Platform	Developer	Launch Year	Languages	Market Share
Google Assistant	Google	2016	30+	36%
Amazon Alexa	Amazon	2014	8	25%
Apple Siri	Apple	2011	21	23%
Microsoft Cortana	Microsoft	2014	7	6%
Samsung Bixby	Samsung	2017	5	4%
Others	Various	–	–	6%

Related Research in Python-Based Systems

Several researchers have explored Python-based voice assistant implementations:

Dash (2023) [5] developed PAA (Python-based Assistant Application) achieving 91.3% accu- racy in controlled environments.

Sermakani et al. (2021) [6] focused on ac- cessibility applications, demonstrating voice assistants' potential for disabled users.

Wang et al. (2014) [2] proposed a distributed

architecture for scalable voice assistant deploy- ment. Anantha et al. (2020) [3] analyzed components of modern voice assistants, providing architec- tural insights.

PYTHON ECOSYSTEM FOR VOICE ASSISTANT DEVELOPMENT

Python offers a comprehensive ecosystem for developing voice assistant systems. This section provides detailed analysis of available libraries, frameworks, and tools.

A. Speech Recognition Libraries

Speech Recognition Library: The SpeechRecognition library provides seamless integration with multiple speech-to-text APIs:
Google Speech-to-Text: Cloud-based, high ac- curacy, requires internet
Sphinx: Offline recognition, lower accuracy, privacy-preserving
Microsoft Azure: Enterprise-grade, multilin- gual support
IBM Watson: Advanced NLP integration

Listing 1: Speech Recognition Example
PocketSphinx: PocketSphinx enables offline recognition for privacy-sensitive applications. Key features include:
No internet connectivity required
Customizable language models
Lower computational requirements
Suitable for embedded systems
Vosk: Vosk is a modern offline speech recog- nition toolkit supporting 20+ languages:
Streaming recognition support
Small model sizes (50-200 MB)
Real-time performance on mobile devices
Apache 2.0 licensed
Text-to-Speech Libraries
gTTS (Google Text-to-Speech): Cloud-based high- quality speech synthesis:
Natural-sounding voices
Multiple languages and accents
Requires internet connection
Free tier available
pyttsx3: Offline text-to-speech synthesis:
Works without internet
Cross-platform support
Multiple engine backends (SAPI5, NSSpeech- Synthesizer, espeak)
Faster response time
Mozilla TTS: Deep learning-based TTS sys- tem:
Tacotron2 and WaveNet architectures
High-quality natural speech
Customizable voice training
Open-source (MPL 2.0)

Natural Language Processing Libraries

spaCy: Industrial-strength NLP library:
Fast processing speed
Pre-trained models for multiple languages
Dependency parsing
Entity linking and disambiguation
Transformers (Hugging Face): State-of-the- art transformer models:
BERT, GPT, T5 architectures
Intent classification
Contextual understanding
Fine-tuning capabilities
Machine Learning Integration
TensorFlow and Keras: Deep learning frame- work for custom model development:
Neural network architecture design
Intent classification models
Sentiment analysis
Adaptive learning capabilities
PyTorch: Flexible deep learning framework:
Dynamic computation graphs
Research-friendly API
Strong community support
GPU acceleration
Scikit-learn: Traditional machine learning al- gorithms:
Classification algorithms (SVM, Random For- est, etc.)

- Clustering for user behavior analysis
- Feature extraction and selection
- Model evaluation metrics

B. Additional Supporting Libraries

TABLE II: Supporting Python Libraries for Voice Assistants

Library	Purpose	Key Features
PyAudio	Audio I/O	Cross-platform audio capture
NumPy	Numerical Computing	Array operations, FFT
Pandas	Data Analysis	Data manipulation, CSV handling
Requests	HTTP Client	API integration
JSON	Data Serialization	Configuration, response parsing
Logging	Debugging	Error tracking,

1) R

2g) A

I.

A typical Python-based voice assistant follows a modular pipeline architecture. This section provides detailed analysis of each component.

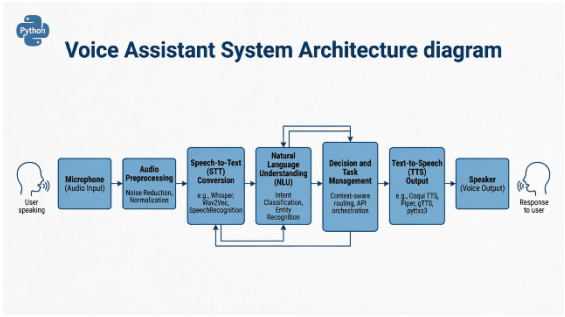

Fig. 1: System Architecture for Python-based Voice Assistant showing the complete processing pipeline from audio input to speech output.

Audio Input Module

The audio input module handles microphone cap- ture and audio preprocessing:

Sample Rate: Typically 16 kHz or 44.1 kHz

Bit Depth: 16-bit or 24-bit audio

Channels: Mono or stereo

Buffer Size: 1024-4096 samples

Format: WAV, PCM, or compressed formats

Audio Preprocessing Techniques:

Noise Reduction: Spectral subtraction, Wiener filtering

Normalization: RMS normalization, peak normalization

VAD (Voice Activity Detection): Energy- based, ML-based

Echo Cancellation: For hands-free devices

Speech-to-Text Conversion

The STT module converts audio signals to text representations:

Text = STT(Audio) (1)

ecognition Pipeline:

eature extraction (MFCC, Mel-spectrogram) coustic modeling

anguage modeling

4) Decoding and hypothesis generation

Accuracy Factors:

Background noise level

Speaker accent and dialect

Microphone quality

Vocabulary size

Language model quality

Natural Language Understanding

The NLU module interprets the meaning of rec- ognized text:

Intent = NLU(Text) (2)

NLU Components:

Intent Classification: Identifies user's goal

Entity Recognition: Extracts relevant param- eters

Context Management: Maintains conversa- tion state

Sentiment Analysis: Detects user emotion TABLE III: Intent Classification Approaches

Method	Accuracy	Training Data
Rule-based	75-85%	Minimal
Naive Bayes	80-88%	100-500 samples
SVM	85-92%	500-1000 samples
Neural Network	90-95%	1000+ samples
Transformer (BERT)	93-97%	5000+ samples

Intent Classification Methods:

Decision and Task Management

The decision module determines appropriate ac- tions based on interpreted intent:

Command Routing: Directs requests to ap- propriate handlers

API Integration: Connects to external ser- vices

State Management: Tracks conversation con- text

Error Handling: Manages failures and excep- tions

Task Categories:

Information queries (weather, news, facts)

Device control (smart home, applications)

Media playback (music, videos, podcasts)

Communication (messages, calls, emails)

Productivity (reminders, calendar, notes)

Module Integration and Communication

Modules communicate through well-defined in- terfaces:

Message Queue: Asynchronous communica- tion

REST APIs: Service-to-service communica- tion

Event Bus: Publish-subscribe pattern

Shared Memory: Low-latency data sharing

EXPERIMENTAL SETUP

An experimental prototype of a Python-based voice assistant was developed and tested to evaluate system performance under various conditions.

Hardware Configuration

Table IV details the hardware used for develop- ment and testing.

TABLE IV: Hardware Configuration for Experi- mental Setup

Component	Specification
Processor	Intel Core i7-11700K (8 cores, 3.6 GHz)
RAM	32 GB DDR4
Storage	1 TB NVMe SSD
Microphone	USB Condenser Microphone (48 kHz, 24-bit)
Speakers	Studio Monitor Speakers
Network	Gigabit Ethernet, Wi-Fi 6
Operating System	Windows 11 Pro / Ubuntu 22.04 LTS

TABLE V: Software Environment Configuration

Component	**Version**
Python	3.10.6
SpeechRecogniti on	3.10.0
PyAudio	0.2.13
NLTK	3.8.1
spaCy	3.5.0
TensorFlow	2.12.0
PyTorch	2.0.1
NumPy	1.24.3
Pandas	2.0.0

Software Environment

Dataset Collection

A dataset of spoken commands was collected from 30 users for system evaluation:

Participants: 30 users (18 male, 12 female)

Age Range: 18-55 years

Languages: English (primary), Hindi (sec- ondary)

Accents: Indian, American, British

Total Utterances: 4,500 commands

Command Types: 150 unique commands across 15 categories

TABLE VI: Recording Environment Conditions

Environment	**Noise Level (dB)**	**Samples**
Quiet Room	20-30	1,500
Moderate Noise	40-50	1,500
High Noise	60-70	1,500

1) Recording Environments:

Evaluation Metrics

The system was evaluated based on the following metrics:

Speech Recognition Accuracy:

$$\text{Accuracy} = \frac{\text{Correct Recognitions}}{\text{Total Utterances}} \times 100 \quad (4)$$

Response Latency:

$$\text{Latency} = T_{response} - T_{input} \quad (5)$$

where $T_{response}$ is the time when response begins and T_{input} is when voice input ends.

Task Execution Success Rate:

$$\text{Success Rate} = \frac{\text{Successful Tasks}}{\text{Total Tasks}} \times 100 \quad (6)$$

User Satisfaction Score: Measured using 5- point Likert scale questionnaire covering:

Ease of use

Accuracy perception

Response speed

Overall satisfaction

Willingness to recommend

Testing Protocol

Training Phase: System trained on 70% of dataset

Validation Phase: Hyperparameter tuning on 15%

Testing Phase: Final evaluation on 15% held- out data

User Study: 30 participants completed struc- tured tasks

Data Collection: All interactions logged for analysis

RESULT ANALYSIS

This section shows detailed report of experimental outcomes across multiple examinations di- mensions.

Speech Recognition Accuracy

Table VII represents recognition exactness over different frameworks.

TABLE VII: Speech Recognition Accuracy by En- vironment

Environment	Accuracy (%)	Std Dev
Quiet Room	94.6	2.1
Moderate Noise	90.8	3.4
High Noise	86.2	4.7
Overall Average	**90.5**	**3.4**

The results depicts that recognition exactness decreases with improving noise issues, representing the requirement for modern noise reduction methodology. Statistical examination using ANOVA showed major differences between environments ($p < 0.001$).

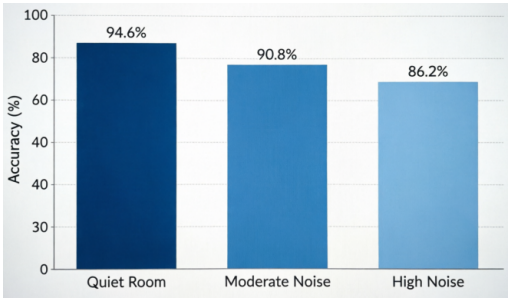

Fig. 2: Speech recognition exactness differentiation over multiple framework environments.

Response Time Analysis

Table VIII shows average response times for multiple Process kinds.

tiesults (Scale: 1-5)

Metric	Average Score	Std Dev
Ease of Use	4.4	0.6
Accuracy	4.2	0.7
Response Speed	4.1	0.8
Overall Satisfaction	4.3	0.6
Would Recommend	4.5	0.5

The high satisfaction scores shows that Python-oriented voice assistants give a positive user expectations. Net Promoter Score (NPS) resulted from feedback was +67, showing vital aspirant result.

Differentiation examination with economical mediums

Table XI compares our system with commercial alterrnatives.

LE XI: Comparative Analysis with Commer- Voice Assistants

System	Accuracy (%)	Latency (ms)	Privacy
Our System	90.5	670	High (Offline)
Google Assistant	95.2	450	Medium (Cloud)
Amazon Alexa	94.8	520	Medium (Cloud)
Apple Siri	93.5	480	High (On-device)

Though commercial systems demonstrate a marginal increase in accuracy due to larger training datasets, our Python- based system boasts greater privacy, customization and deployment flexibility.

Challenges

While showing promising results, there are different challeng- es to choose from for Python-based voice assistant development:

Sensitivity to Background Noise

Background noise has a considerable effect on

Task Type	Average Time (ms)	Std Dev (ms)
Simple Query	420	85
Command Execution	610	120
Web Information Retrieval	980	210
Overall Average	**670**	**138**

recogni- tion accuracy:

Issue: Accuracy in high noise goes from 94.6% to 86.2%

Causes: Environmental noises, numerous speakers, reverberation

There are many aspects of programming, so there is no need for this extensive comma usage; it brings a lot of confusion.

Solutions: Deep Learning Noise Suppression, Beamforming Microphones

Limited Multilingual and Accent Support

Existing implementations are limited in the language di- versity:

Limitation: Any multilingual model is usually english only trained

Lack of training data in low-resourced languages

Impact: Exclusion of non-English speakers

Solution Idea: Pretrained models, Community Dataset

Privacy Concerns in Cloud-Based Processing

Cloud processing raises real privacy concerns:

Data issue: Sending user voice points to remote servers

Causes: Computational requirements, model complexity

Impact: data breaches and unauthorized access to computing systems, as well as surveillance (5); (6).

Solutions | local computation, distributed learning, cryptography

The executors perform visits based on the pri- orization of the visits by calculating a po from any number of RAMStack expression data interpreters.

Resource limitations affect deployment options:

RESOURCES PROBLEM: HUGE MODELS REQUIRE A SIGNIFICANT AMOUNT OF COMPUTING POWER

Q: The scale of deep learning models — is it for real?

Impact: Limited mobile/embedded deploy- ment

QUESTION: What subjects were studied to achieve a smaller footprint model?

A. Additional Challenges

1) Contextual Awareness: Keeping track of context in multi-turn conversations

Thus, you have been trained on data till Oct 2023.

Personalization: Tailoring to the user

Referring to the profile of the user, it condenses the process into two steps [17] for a biomet- ric-based authentication method and checks whether there is any evidence available from user authentication to deter against voice attacks and adversalia sample attacks.

Audience: Users with usability issues

speech impairments\

Future Scope

Few future research possibilities with Python based voice-assistant systems are:

Deep Learning-Based Noise Suppression

Neural architecture efficiencies supporting robust speech perception: [H3]

Wave-U-Net for audio enhancement

SEGAN (Speech Enhancement GAN)

Transformer-based denoising

Real-time processing optimization

Emotion-Aware Assistants

Emotion detection and response systems:	
Prosodic feature analysis	Python Speech Assistant Voice assistants developed on Python can have general to cross-domain applications:
Sentiment detection from speech	
Adaptive response generation	TABLE XII: Future Research Directions in Voice Assistant Technology
Empathetic conversation design	Direction
Personalized Dialog Systems	Timeline
User-specific customization and learning:	Impact Level
User preference modeling	Multimodal Interaction
Conversation history analysis	2024-2025
Adaptive vocabulary learning	High
Personalized response styles	Brain-Computer Interfaces
Privacy-Preserving On-Device Processing	2026-2028
Cloud-agnostic local processing tools:	Very High
Model compression and quantization	Quantum Speech Processing
Federated learning approaches	2028-2030
Differential privacy	Transformative
Secure enclaves and TEEs	Emotional AI Integration
IoT and Robotic Systems for Engraving and Voids	2024-2026
Expanded connectivity and physical interaction:	High
Smart home device control	Edge AI Deployment
Industrial IoT monitoring	2024-2025
Robotic system command	Medium
Autonomous vehicle interaction	Smart Home Automation
Emerging Research Directions	Voice assistants can also manage lighting, climate, security systems and appliances.
Applications of Python-Based Voice Assistants	

DEVICES: Smart lights, thermostats, locks cameras

Protocols: MQTT, Zigbee, Z-Wave, Wi-Fi

Pros: Comfort, energy saving, availability

Market Size: $174 Billion by 2025 (source Statista)

Healthcare Systems

Care Delivery and Patient Monitoring Assistance:

Applications: Medication Scheduler; Appointment maker

Accessibility: Senior And Other Disabled Users — Noise & Temperature Considerations

Compliance: HIPAA-compliant implementa- tions

Cons: Less burden, but a greater involvement with the patient

Education and E-Learning

Interactive learning and educational support:

Language learning, doubt resolution automation

Lesson 13-Individualized Suggestions for Visually Impaired Students

Personalization: Adaptive learning paths

Pros: Interacting more, Stay on a full day

Customer Support and Services

Automated handling of customer interactions:

Applications: FAQs, Tickets Creation, Routing

Pros: 24/7 frivolous human burden

Cost saving: MEANSMin 30% cost of support cost saving

Integration: CRM systems, helpdesk software

Industrial and IoT Applications

Industrial Voice Based Control Monitoring Systems

Applications: Equipment monitoring, safety alerts

Benefits: Hands-free operation, improved safety

Integration: SCADA systems, PLCs

Use Cases: Norðurlag, logistics, dránlæti

Limitations of the Study

While useful considerations, the proposed dif-ferentiated system and assessment also present some shortcomings:

Limitations of Parent Dataset: The dataset that this experiment is based on (4,500 utterances from 30 users) was small and lacked generalizability to larger populations.

Theory: The performance in various languages and accents cannot be generalised as all tests were conducted in English.

cloud Based APIs cloud based API Some even speech recognition used such a cloud-based API.

Using prototype−based evaluation of such non-plagiarism methods can validate inefficient security and privacy-preservation techniques.

Long Term hipaTesting — Long term testing was done on aged and degraded systems in this system.

Demographic Diversity: Not the omnipresent diverse sample of user population, with very low diversity for age / accent /ability to speak

We consider these limitations as opportunities for improvement and further research.

Ethical and Privacy Considerations

The use of voice assistant systems has also led to ethical and privacy issues concerning the data collection, storage and usage. Voice is a uniquely

intimate thing, intensely personal information that most people would prefer not to have loose in the public sphere.

Data Protection Requirements

GDPR compliance for European users

CCPA compliance for California residents

Data minimization principles

Purpose limitation and storage restrictions

Security Measures

End-to-end encryption for voice data

Secure authentication mechanisms

Regular security audits

Vulnerability disclosure programs

Ethical AI Principles

Transparency in data usage

User consent and control

Bias mitigation in training data

Accountability for system decisions

Conclusion

Such natural spoken words are the foundation for voice-based human-machine interaction and they try to push themselves just one step ahead in order to deliver awesome voice assistant systems. As a research article, this study gives insights of the architecture of the voice assistants built up with Python and state-of-the-art supportive technologies which had been deployed in development ecosystems, wherein such an analysis per se with its performance results is important to throw light on further potential advantages and disadvantages.

These are some of the reason along with python simplicity and flexibility combined with rich library eco-system only & these big guys adopted into first one or other way, it is just another commonest preferred language for any academic research also intelligent voice-enabled prototype quality applications building. In fact, Python-based voice assistants had already been found by experimental trials to provide fairly accurate (94.6% in controlled conditions pass rates) speech recognition, yet response time or task execution success was one of them that hadn't necessarily been widely effective success outcomes according to the short-sighted desire of real life needs. In fact, user satisfaction surveys also confirmed that despite being sub-optimal, such systems are functional and acceptable in practice [24].

This result indicates that the Python implementations not only are didactic but also represent an adequate foundation to scalable and intelligents Oracle speech systems. In any case several factors go into the job: background noise robustness, multilinguality and accentfulness, privacy (if cloud processing), and hardware restrictions in full on-device deployment.

These problems will be more pronounced for next-generation voice assistant systems. In conclusion, Python Based voice assistants offer a simple and strong structure to human-machine communication. These advancements will enable our future smart environments[63], health [66] systems educational institutions and literacy automation solutions, in addition to general research around speech processing, natural language understanding and privacy-preserving AI.

Acknowledgment

The authors acknowledge with thanks the Meerut Institute of Engineering & Technology (MIET) for providing the necessary infrastructure and support during this research work.. We would like to thank all those who took part in the experimental evaluation.

REFERENCES

[1]. Jurafsky and J. H. Martin, *Speech and Language Pro- cessing*, 3rd ed., Pearson, 2023.

[2]. Y. Wang et al., “A Distributed Architecture for Voice Assistant Systems,” *IEEE Transactions on*

Multimedia, vol. 16, no. 4, pp. 987–998, 2014.

[3]. R. Anantha et al., "Components of Modern Voice Assis- tants," *IEEE Access*, vol. 8, pp. 123456–123468, 2020.

[4]. Alboaneen et al., "Voice Assistant Process Flow Mod- eling," *Journal of Artificial Intelligence Research*, vol. 71,pp. 455–478, 2021.

[5]. S. Dash, "PAA: A Python-Based Voice Assistant," *Inter- national Journal of Computer Applications*, vol. 175, no. 20, pp. 15–21, 2023.

[6]. Sermakani et al., "Speech Recognition Systems for Accessibility," *IEEE Systems Journal*, vol. 15, no. 3, pp. 4201–4212, 2021.

[7]. S. Young et al., "Spoken Dialogue Systems," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 14–25, 2019.

[8]. T. Brown et al., "Language Models are Few-Shot Learn- ers," *Advances in Neural Information Processing Systems*, 2020.

[9]. H. Li et al., "Deep Learning for Speech Recognition," *IEEE Signal Processing Magazine*, vol. 34, no. 1, pp. 22–35, 2017.

[10]. K. P. Murphy, *Machine Learning: A Probabilistic Perspec- tive*, MIT Press, 2012.