INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 4, April 2026 | ISSN: 2456-4184 | IJNRD.ORG

ORION AI: A Python Based Intelligent Chatbot with NLP and Speech Recognition

Authors:
Abhishek Saxena Anshik Sharma Ashutosh Divedi Bhanu Pratap Pawar Guide: Mr. Lav Kumar Dixit
Department: Computer Science and Engineering

Abstract

Natural language interfaces are becoming an important part of modern human-computer interaction because they allow users to control systems in a more intuitive way. This project presents a voice-driven desktop automation system that converts spoken commands into structured JSON instructions using a ChatGPT-based model. The generated JSON is then passed to a Python backend, which validates the instruction and executes the corresponding terminal action. The system supports common tasks such as opening websites, launching applications, and searching videos on YouTube through voice input.

The workflow combines speech recognition, large language model-based semantic interpretation, and command execution in a modular pipeline. Unlike rule-based assistants that depend on fixed phrases, this approach is designed to understand flexible user language and convert it into a machine-readable format. OpenAI's structured output capability is especially useful in this context because it helps ensure that the model returns valid JSON according to a defined schema. Speech recognition is handled through a Python library that supports multiple engines and APIs, while Flask can be used to provide a lightweight web interface for interaction and status display.

1. Introduction

Voice-based interaction is one of the most natural ways for humans to communicate with digital systems. Traditional command-based automation tools usually require users to remember exact syntax or predefined phrases, which reduces usability and flexibility. In contrast, semantic parsing techniques aim to convert natural language into executable representations, making it possible for systems to understand user intent more directly. Research in command understanding has shown that natural language instructions can be mapped into structured forms for execution, which is closely aligned with the idea behind this project.

In this project, the goal is to build a practical voice automation system that listens to a spoken command, converts it into JSON using a ChatGPT model, and then executes the resulting action through the terminal. This makes the system suitable for desktop automation tasks such as opening YouTube, searching for a video, or launching an application. The project also reflects the growing capability of modern speech-enabled large language models, which now support speech modality interaction and structured responses in practical assistant workflows.

2. Problem Statement

Most traditional desktop automation systems are not flexible enough to understand natural spoken language. They usually require predefined command formats, which limits usability for non-technical users. In addition, many systems can interpret input but cannot reliably convert that input into a safe, structured, machine-readable format before execution.

The main problem addressed in this project is how to design a system that can: understand spoken

instructions,
convert them into valid JSON,
validate the JSON,
and safely execute the intended terminal command.

3. Existing System

Existing automation tools often depend on fixed command rules, macros, or keyword matching. While these methods can work for simple tasks, they struggle when the user speaks naturally or uses different phrasing for the same request. They also lack a clean intermediate format for validation, which can make execution brittle and harder to secure.

Some voice assistants use structured intent recognition pipelines, where speech is transcribed first and then converted into a structured JSON-like event. The voice2json approach is one example of this type of pipeline, where transcription is passed to an intent recognizer and the final result becomes a structured JSON event.

4. Proposed System

The proposed system uses ChatGPT as the language understanding core of the assistant. The workflow begins when the user speaks a command. The audio is captured and converted into text using speech recognition. The transcribed text is then sent to the ChatGPT model with instructions to return only valid JSON. The JSON contains fields such as action, target, and query, depending on the user's intent.

After receiving the JSON, the Python backend validates the structure and checks whether the requested action is allowed. If the instruction passes validation, the system dispatches the command to the terminal or browser. For example, a command such as "open YouTube and
search for AI videos" can be converted into a JSON instruction that launches a browser search automatically.

This design separates interpretation from execution, which improves reliability and makes the system easier to extend.

5. Tools and Technologies Used Component
Technology Programming Language Python
AI Model ChatGPT / OpenAI API
Speech Input SpeechRecognition library Web
Framework Flask
Database MySQL
Frontend HTML, CSS, JavaScript
Command Execution Python subprocess, os, browser automation tools if used

SpeechRecognition is a Python library for speech recognition and supports several engines and APIs. Flask is a lightweight WSGI web framework that is commonly used for building small-to-medium web applications. MySQL is an open-source relational database management system used for storing structured data. OpenAI's structured outputs feature is useful here because it helps ensure the model returns output that follows a supplied JSON schema.

6. System Implementation

The system is implemented as a modular pipeline.

First, the voice input is captured through a microphone and converted to text by the speech recognition module. Second, the transcribed text is sent to the ChatGPT model with a strict instruction to return only JSON. Third, the backend parses the JSON and verifies that it contains an approved action. Fourth, the system executes the

corresponding terminal command or browser operation.

A typical JSON structure may look like this:

```
{
 "action": "search_youtube", "query": "latest AI
 tutorials"
}
```

or

```
{
 "action": "open_website", "target": "youtube"
}
```

This structured format is important because it allows the program to separate user intent from execution logic. It also reduces ambiguity compared with direct natural language execution.

7. Results and Discussion

The system was tested with different spoken commands related to browser opening, video search, and general desktop automation. The output showed that voice input could be successfully converted into structured JSON and then executed through the terminal workflow.

The use of JSON as an intermediate representation improved control over the execution process because the backend could validate the action before running it. This makes the system safer and more dependable than a direct command-execution approach. The design also makes future expansion easier, since additional actions can be added by extending the JSON schema and dispatcher logic.

8. Conclusion

This project demonstrates a practical method for voice-based desktop automation using ChatGPT, JSON parsing, and Python. By combining speech recognition with a large language model, the system can interpret flexible spoken language and convert it into structured instructions for execution. The architecture is modular, extensible, and suitable for common automation tasks such as opening websites and searching videos.

The project also shows the value of structured output in AI-based systems. Instead of letting the model produce free-form text, enforcing JSON output creates a cleaner bridge between human language and machine action. This makes the assistant more reliable and easier to maintain.

9. Future Scope

Future versions of the system can include text-to-speech feedback, stronger command validation, multi-step automation, and support for more operating systems. The system can also be enhanced with a more detailed permission layer so that sensitive commands require confirmation before execution. Multilingual support and offline speech recognition are also practical extensions.

10. References

OpenAI API Documentation: Structured Outputs. SpeechRecognition Library Documentation.Flask Official Documentation.
MySQL Official Documentation / Oracle MySQL overview. Voice2json Whitepaper.
Neural Semantic Parsing for Command Understanding. SOVA-Bench / speech-enabled LLM assistant research.