Design and Implementation of a Desktop Voice Assistant Using Python

Gurdeep Kaur[1], Manu Sharma[2], Mr. Ajeet Singh[3]
[1,2]*Master of Computer Applications (MCA), IV (Final Year), Kanpur Institute of Technology*
[3]*Guide, Kanpur Institute of Technology*

Abstract- A desktop voice assistant is a software application that accepts spoken commands from a user and performs tasks through speech recognition and text-to-speech output. This paper presents the design and implementation of a simple desktop voice assistant named Jarvis using Python. The assistant can greet the user, recognize voice commands, open websites, search Wikipedia, play music, send email, and open applications. The proposed system is lightweight, user-friendly, and demonstrates how Python libraries such as speech recognition, text-to-speech, and web automation can be integrated into a real-world application.

Keywords: Voice Assistant, Python, Speech Recognition, Text-to-Speech, Automation, Desktop Assistant

I. INTRODUCTION

Voice assistants are becoming an important part of modern computing systems. They allow users to interact with machines using natural language instead of traditional input devices like keyboard and mouse. This project focuses on building a simple desktop-based voice assistant using Python that can perform various tasks such as opening applications, searching information, and sending emails.

II. PROBLEM STATEMENT

Many computer tasks require manual interaction using keyboard and mouse. This may be time-consuming for users who want quick access to information or system functions. A voice assistant can reduce this effort by allowing users to control the computer using spoken commands. However, advanced assistants are often complex and require large-scale systems. Therefore, the problem addressed in this project is how to build a small but effective desktop voice assistant using commonly available Python tools.

III. OBJECTIVES

The main objectives of the project are:

1. To design a voice-controlled desktop assistant using Python.
2. To convert spoken language into text using speech recognition.
3. To convert text responses into speech using a text-to-speech engine.
4. To automate common desktop tasks such as opening websites and applications.
5. To provide a simple and interactive user experience.

IV. SCOPE OF THE PROJECT

This project focuses on a basic desktop assistant with limited but useful features. It is not intended to replace advanced AI assistants like Siri or Alexa. Instead, it serves as a mini assistant for academic demonstration and learning purposes. The assistant can handle fixed commands such as opening Google, YouTube, Wikipedia search, playing music, and sending email. The system can also be extended in the future with more intelligent features.

V. LITERATURE REVIEW

Voice assistants have become an important topic in human-computer interaction. Existing assistants use natural language processing, speech recognition, and machine learning to understand user intent. Research in this area shows that voice interfaces can improve

accessibility, productivity, and convenience. Earlier systems were mostly rule-based and used fixed command patterns. Modern assistants are more advanced and can understand conversational input. However, for student-level projects, a rule-based or hybrid system is often more practical. It allows the developer to focus on core concepts such as input capture, command matching, and output generation. This project follows a simple rule-based approach, which is appropriate for a desktop assistant because it is easy to understand, easy to implement, and suitable for demonstration in an academic environment.

VI. MAJOR FEATURES

The assistant supports these tasks:

- Greeting the user based on time of day
- Opening Google and YouTube
- Searching Wikipedia
- Playing songs from a local folder
- Opening Visual Studio Code
- Sending emails through SMTP
- Speaking responses using text-to-speech

VII. SYSTEM REQUIREMENTS

7.1 Hardware Requirements

- A laptop or desktop computer
- Microphone
- Speakers or headphones
- Minimum 4 GB RAM recommended

7.2 Software Requirements

- Python 3.x
- Windows operating system
- Python packages:
  - speech_recognition
  - pyttsx3
  - pyaudio
  - wikipedia
  - webbrowser
  - smtplib
  - os
  - datetime
  - time

VIII. TOOLS AND TECHNOLOGIES USED

8.1 Python

Python is used as the main programming language because it is simple, readable, and supports many useful libraries.

8.2 SpeechRecognition

This library is used to capture voice input from the microphone and convert it into text.

8.3 pyttsx3

This library is used for text-to-speech output so the assistant can speak to the user.

8.4 Wikipedia Library

This library helps fetch short summaries from Wikipedia.

8.5 webbrowser Module

This module opens websites such as Google and YouTube in the default browser.

8.6 smtplib

This module is used to send emails through SMTP.

8.7 os Module

This module is used to access local files and open applications or music files.

IX. METHODOLOGY

The assistant follows a simple command-based workflow.

Step 1: Voice Input

The program listens to the user through the microphone using the SpeechRecognition library.

Step 2: Speech to Text Conversion

The spoken input is converted to text by Google Speech Recognition.

Step 3: Command Matching

The system checks whether the text contains keywords such as “wikipedia,” “google,” “youtube,” or “email.”

Step 4: Task Execution

If a command matches, the assistant performs the corresponding task.

Step 5: Text-to-Speech Output
The assistant gives a spoken response using pyttsx3.

X. FLOW OF THE SYSTEM

The working flow of the project can be summarized as:

User speaks → microphone captures audio → speech recognition converts it to text → command is matched → task is executed → assistant responds by voice

XI. IMPLEMENTATION DETAILS

The project is implemented in Python using a main loop that continuously listens for commands. Each command is handled using if-elif conditions. The assistant is implemented using Python and uses various libraries:

- SpeechRecognition for voice input
- pyttsx3 for speech output
- Wikipedia API for information retrieval
- SMTP for email sending
- OS module for file handling

11.1 Greeting Function
The assistant greets the user depending on the current time.

11.2 Wikipedia Search
When the user says a command containing "wikipedia," the assistant removes the keyword and fetches a brief summary from Wikipedia.

11.3 Open Google
When the user says "open google," the assistant opens Google in a browser.

11.4 Open YouTube
When the user says "open youtube," the assistant opens YouTube.

11.5 Play Music
The assistant reads songs from a local folder and opens a selected audio file.

11.6 Send Email
The assistant can send an email using SMTP and an app password.

XII. SAMPLE WORKING COMMANDS

Some example commands supported by the system are:

- "open google"
- "open youtube"
- "search Wikipedia for Python"
- "play music"
- "open code"
- "send email"

XIII. RESULTS AND DISCUSSION

The implemented voice assistant successfully performs the selected desktop tasks. The assistant is able to listen to commands, recognize text input, and execute actions based on the spoken instruction. It also provides audio responses using speech synthesis. During testing, the assistant worked well for basic commands such as opening websites and fetching Wikipedia summaries. Some issues were observed during development, such as microphone overflow, path errors, and Gmail authentication restrictions. These problems were solved by improving the command logic, cleaning the text input, using correct file paths, and enabling app password authentication for email. The final system is stable enough for academic demonstration and clearly shows how voice-based automation can be implemented in Python.

XIV. ADVANTAGES OF THE PROPOSED SYSTEM

- Easy to use through voice commands
- Saves time compared to manual interaction
- Helpful for learning speech-based automation
- Can be extended with more smart features
- Suitable for beginners and academic projects

XV. LIMITATIONS

- Works only for predefined commands
- Depends on microphone quality and internet for speech recognition

- Limited natural language understanding
- Email sending requires proper Gmail security setup
- Not as advanced as commercial AI assistants

XVI. FUTURE SCOPE

The project can be improved in the future by adding:

- Wake word detection such as “Hey Jarvis”
- Better natural language processing
- GUI interface using Tkinter or PyQt
- Chat history and memory
- More smart home or PC automation features
- Offline speech recognition support

XVII. CONCLUSION

A desktop voice assistant is a practical example of human-computer interaction using speech recognition and speech synthesis. In this project, a simple Python-based assistant named Jarvis was designed and implemented to perform daily computer tasks through voice commands. The system demonstrates how a small but effective voice assistant can be built using accessible tools and libraries. The project is useful for academic learning because it combines programming, automation, and basic AI-related concepts. It also provides a strong foundation for future work in intelligent assistants and voice-enabled applications.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide Mr. Ajeet Singh for his valuable guidance, support, and encouragement throughout the development of this project. We are also thankful to Kanpur Institute of Technology for providing us with the opportunity and resources to complete this project successfully. We would like to thank our faculty members, friends, and family for their constant support and motivation.

REFERENCES

[1] Python Official Documentation
[2] SpeechRecognition Library Docs
[3] pyttsx3 Documentation
[4] Wikipedia API and Python Library Documentation
[5] SMTP and smtplib Protocol Documentation

APPENDIX (COMMANDS)

A voice-enabled desktop assistant that recognizes spoken commands and performs tasks such as

- opening websites
- open google
- open youtube
- play music
- search wikipedia (according to wikipedia)
- send email
- open code
- the time