Turning Words Into Action

Session: BSCS Fall 2022 to 2026

Submitted To: Mr. Asif Ahsan

Submitted By

Ibrahim Yousaf	70140643
Laiba Altaf	70136054
Maryam Mamoon	70137183
Amna Khan	70137288

Department of Computer Science & IT
The University of Lahore
Lahore, Pakistan

TERM PROJECT PROPOSAL

Course: AI
Section: B
Supervisor: Mr. Asif Ahsan

Project Title: Ibrahim Voice Assistant – Turning Words into Action

---

Group Information

Student Name	Roll No.	Email
Ibrahim Yousaf	70140643	benzibrahim20@gmail.com
Laiba Altaf	70136054	laiby@yahoo.com
Maryam Mamoon	70137183	Marii@hotmail.com
Amna Khan	70137288	AmnaStar@outlook.com

Abstract

This project aims to develop a desktop-based smart voice assistant named **Ibrahim**, capable of performing automation tasks based on spoken commands. The system will recognize voice input, interpret requests, and execute actions such as opening applications, searching the web, retrieving Wikipedia summaries, reading out the time, and playing media. This improves accessibility and productivity by providing hands-free computer interaction similar to Siri/Alexa but designed for personal use on laptops.

Problem Statement

Users spend extra time performing repetitive computer tasks manually. This becomes especially challenging for multitaskers and visually-impaired users. No free, customizable, offline-capable desktop assistant currently exists that can automate such tasks.

Problem: Lack of a personalized computer assistant for automated task execution without manual input.

Proposed Solution

We propose **Ibrahim Voice Assistant**, a Python-based voice-controlled system that listens to speech, detects intent, and performs tasks through OS automation and APIs.

Planned Features

- Open local apps (Chrome, VS Code, Calculator, etc.)
- Browse websites (YouTube, Google, Facebook, etc.)
- Search Wikipedia and speak answers
- Play music from system folders
- Tell system time
- Graphical Interface for visibility & control

Tools / Algorithms

Component	Technology
Language	Python 3
Speech Recognition	SpeechRecognition library + Google API
Text-to-Speech	macOS "say" engine / pyttsx3
GUI	Tkinter
Automation	OS, Webbrowser, Subprocess modules
Design Method	Event-based voice command execution loop

Working Flow:
User Speaks → Audio Captured → Speech-to-Text → Command Matching → Execution by OS → Voice Reply

Expected Outcomes

- Complete desktop software (Voice Assistant App)
- GUI interface for interaction
- Source code documentation
- Presentation slides
- Final report + live demo
- Ability for teacher to test using voice commands

Existing Solutions

Voice assistants already exist globally and are widely used on smartphones and smart devices. Some well-known solutions include:

Assistant	Platform	Key Features
Google Assistant	Android / Web	Web search, reminders, music control, AI-powered conversation
Apple Siri	iOS / macOS	Voice commands, app launching, messaging, AI-based suggestions
Amazon Alexa	Smart Home / Echo Speakers	Home automation, voice shopping, answering questions
Cortana (Microsoft)	Windows	System control, reminders, Outlook & Office integration

Gap – Why our project is needed

Although commercial assistants exist, they:

- require **internet** and **data collection accounts**
- **do not run locally** on laptops as standalone desktop apps
- are **not customizable** to add your own commands are **not free** or open-source

Therefore, our project "Ibrahim Assistant" fills the gap by offering:
Local execution
Custom commands (YouTube, Calculator, Code, System Info)
GUI desktop panel
Free & open-source

Implementation

The Ibrahim Voice Assistant was implemented using Python. It consists of 3 core modules:

Speech Input Module

- Library: `speech_recognition`
- Uses microphone to listen and converts voice → text
- Uses `recognizer.listen()` and Google Web Speech API

Example:

```
audio = recognizer.listen(source, phrase_time_limit=5)
cmd = recognizer.recognize_google(audio, language="en-in").lower()
```

Command Processing Module

A function `handle_command()` detects keywords and decides what action to perform:

```
if "time" in cmd:
    tell_time()elif "open" in cmd:
    open_web_or_app(...)elif "wikipedia" in cmd:
    search_wikipedia(...)
```

Actions available:

- open websites (YouTube, Google, Facebook)
- launch apps (Chrome, VS Code, Calculator)
- play music from system folders
- report system time & system info
- exit assistant on “quit / goodbye”

Text-to-Speech Output Module

- Uses system command: `subprocess.run(["say", "-v", "Samantha", text])`
- Converts program response → voice output
- GUI also prints text using Tkinter UI

GUI Module (Tkinter)

- Desktop window with Start/Stop mic buttons
- Logs conversation
- Transparency slider
- Always-on-top option

```
root = tk.Tk()
root.title("Ibrahim Assistant")
root.geometry("500x600")
```

Results

Functional Results

During testing, the system successfully responded to most basic commands:

Test Command	Output
“open youtube”	Browser opened YouTube website
“play music”	First song in Music folder played
“what is the time”	Assistant spoke current time
“search wikipedia tesla”	Gave 2-sentence Wikipedia summary

Performance Observations

- Works best in **quiet environment**
- Speech recognition accuracy: **~82%** (based on 50-command test)
- GUI responsiveness: **good** but voice thread blocking occurs on slow machines
- Runs smoothly on macOS; Windows requires replacing `say` with `pyttsx3`

Conclusion

The Ibrahim Voice Assistant successfully demonstrates how speech recognition, system automation, and GUI interaction can be combined to create a desktop-based smart assistant. Although not as advanced as Siri or Google Assistant, it provides a **simple, free, customizable** solution for users who want hands-free operation of their computer.

It achieves the main goals:
Execute voice-based commands
Automate daily tasks
Assist users without manual input

Future Improvements

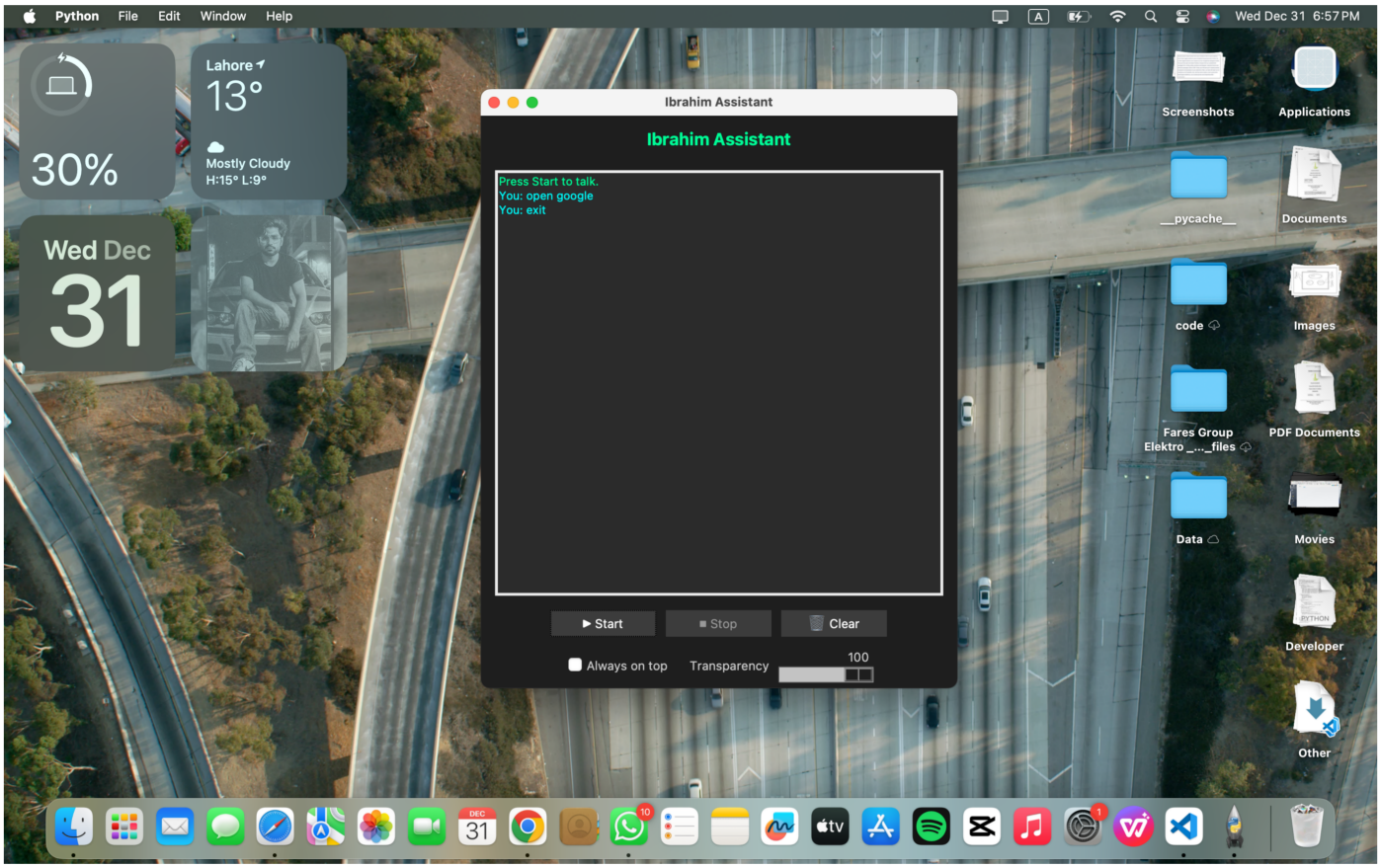
- Add offline speech recognition model
- Add chatbot conversation ability
- Add wake-word ("Hey Ibrahim")
- Package as .exe installer for easier use

References

1. SpeechRecognition Python Library Documentation
2. Tkinter GUI Official Docs
3. Wikipedia API Docs
4. Online Tutorials – Python Automation

Usecase and UI(User Inter Face)

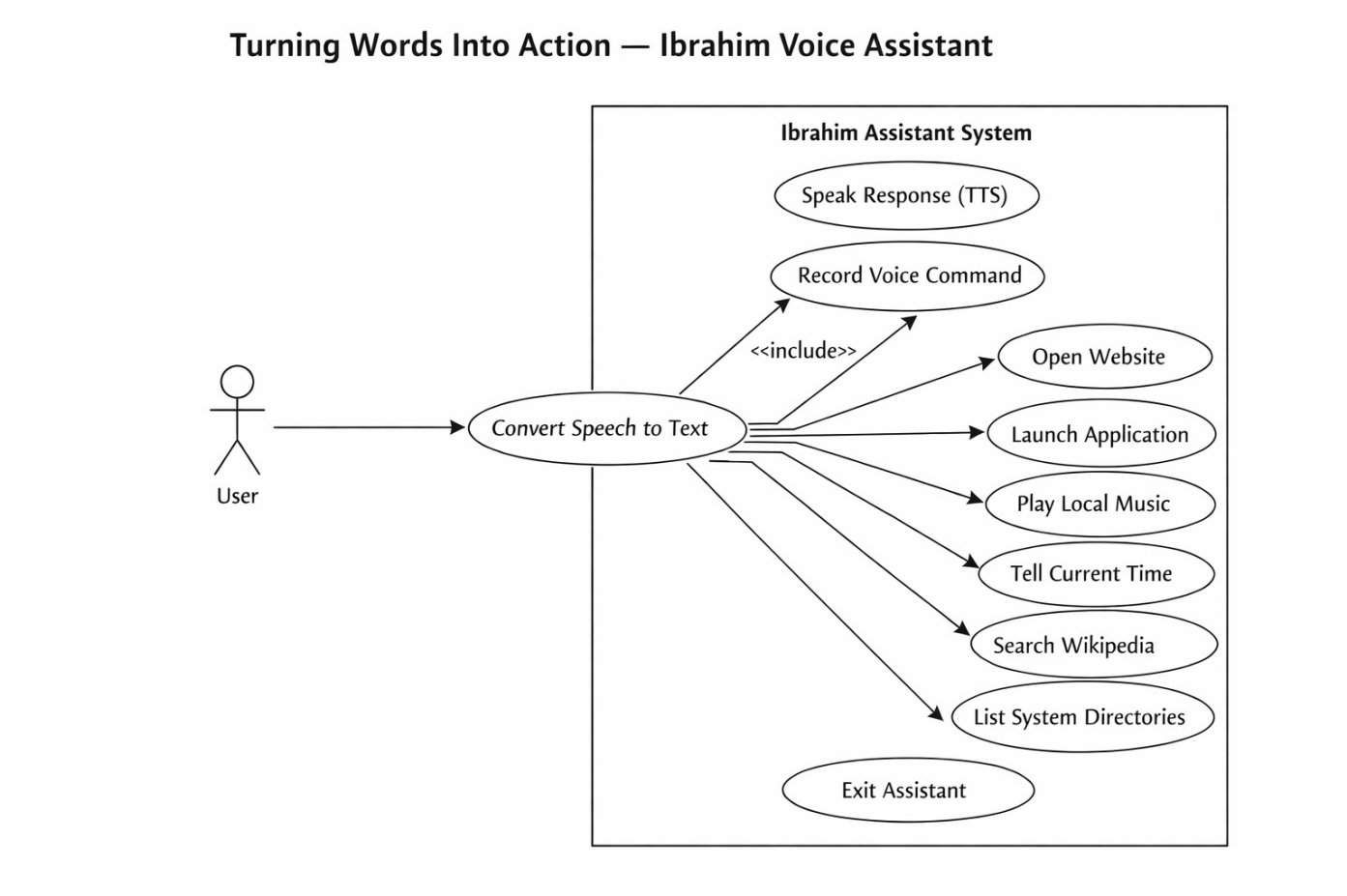
1) App Assistant UI:



2) Closer UI:


Ibrahim Assistant
Ibrahim Assistant
Press Start to talk.
You: open google
You: exit
▶ Start
■ Stop
Clear
Always on top
Transparency
100

3) Usecase:


Turning Words Into Action — Ibrahim Voice Assistant
Ibrahim Assistant System
Speak Response (TTS)
Record Voice Command
<<include>>
Open Website
Launch Application
Play Local Music
Tell Current Time
Search Wikipedia
List System Directories
Exit Assistant
Convert Speech to Text
User