Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2025), B.Sc. in CSE (Day)

Virtual Assistant

Course Title:Artificial Intelligence Lab
Course Code: CSE-316
Section: D14

Students Details

Name	ID
Nujhat Tabasum Tisha	221902215
Ashraful Islam Meraj	221902231

Submission Date: 13/05/2025

Course Teacher's Name: Md. Sabbir Hosen Mamun

[For teachers use only: Don't write anything inside this box]

Lab Project Status

Marks:	**Signature:**
Comments:	**Date:**

Chapter 1

Introduction

1.1 Overview

An artificial intelligence software which can perform tasks using voice command is referred to as a Virtual Assistant.The goal of this project is to develop a Python-based virtual assistant which can accept voice command, complete the given task, and respond in verbal form. The assistant can be made to work interactively with the use of several libraries such as SpeechRecognition that enables voice input, pyttsx3 for text to speech output, and pyaudio for use of audio. Furthermore, additional capabilities such as conducting web searches, opening different software, and playing video or audio file can be performed through the use of the API pywhatkit. This project serves to enhance understanding of working with AI, automation and speech processing, thus it is good in practicing advance uses of voice automation technology.

1.2 Motivation

The motivation of creating the virtual assistant is aimed at increasing automation systems further and ease completion of tasks with just a voice command. As AI powered assistants like Alexa and Siri made their debut, voice command technology integrated itself into smart systems. This project intends to allow for task completion such as web searches, setting up reminders, opening applications and others to all be done simply by talking making it easy for users while at the same time taking personalization and engagement to an all new level. Building the assistant leads to improved understanding on speech recognition, natural language processing, and task automation, which gives the user complete control of how they interact with the assistant.

1.3 Problem Definition

1.3.1 Problem Statement

In today's busy world, people like to multitask as much as possible which means they now look for efficient hands-free methods to get things done whether it is searching for information, setting reminders, playing music, or managing smart devices. Current efforts made through the use of online assistants such as Alexa and Siri, while powerful do come with limitations such as concerns for privacy, lack of custom features, and reliance on certain ecosystems. The objective of this project is to create a virtual assistant powered in Python, that understand and respond to voice commands which is personable and easily adaptable. With the combination of speech recognition, text-to-speech synthesis, and automation, the assistant aims to interact with the users' devices through basic voice commands. This approach caters for the gap in open source, customizable, proprietary free virtual assistants.

1.3.2 Complex Engineering Problem

...

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributes	**Explain how to address**
P1: Depth of knowledge required	Developing a virtual assistant requires Python, speech recognition (SpeechRecognition), text-to-speech (pyttsx3), and audio processing (pyaudio). APIs like pywhatkit and AI/NLP enhance interactions, while system integration allows task automation and app management.
P2: Range of conflicting requirements	—-
P3: Depth of analysis required	The analysis requires understanding speech recognition, text-to-speech, and automation for diverse tasks. It also involves optimizing performance, ensuring privacy, and creating smooth, real-time interactions for a user-friendly experience.
P4: Familiarity of issues	—-
P5: Extent of applicable codes	—-
P6: Extent of stakeholder involvement and conflicting requirements	Stakeholders include users and developers, with potential conflicts between customization, privacy, and performance. Balancing features, data security, and fast responses is key to meeting diverse needs.
P7: Interdependence	—-

1.4 Design Goals/Objectives

- Add automatic voice recognition for the user commands.
- Add response function for the assistant with natural, human quality speech.
- Enable multifunction assistant to open applications, remind users, send emails and play video/audio files.
- Add searching functionality on the internet through APIs.
- Enable modification of commands and responses by the users
- Restrict over-dependability on the cloud by allowing privacy measures and offline capabilities.
- Implementable functions should guarantee user experience and be appealing socially and verbally.

1.5 Application

1. **Personal Assistant:** Provides users with the capability to set reminders, schedule appointments, and send emails.
2. **Hands-Free Control:** Voice commands are used to perform actions like playing music, opening applications, and browsing the internet.
3. **Smart Home Integration:** Extendable to controlling smart home appliances such as lights, thermostats, and security systems.
4. **Educational Support:** For students, helps in answering questions, listening, and reading.
5. **Accessibility Aid:** Provides a user with a disability assistance in the form of hands-free voice navigation and task management.
6. **Automated Task Execution:** Uses a voice command to run scripts or applications to boost productivity.
7. **Entertainment:** Provides songs, jokes and current news or weather reports.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

In the modern era of Artificial Intelligence and automation, voice-controlled virtual assistants have become an essential part of smart systems, making human-computer interaction more intuitive and efficient. This project is a Python-based Voice Assistant, inspired by popular assistants like Amazon Alexa and Google Assistant, that can perform various tasks using voice commands. The assistant uses speech recognition to understand user input and text-to-speech (TTS) to respond verbally. It can perform a range of functions such as telling the time, playing songs on YouTube, fetching summaries from Wikipedia, telling jokes, and providing live weather updates using the OpenWeatherMap API.

2.2 Project Details

This project is focuses on creating an interactive, voice-controlled assistant that can respond to user commands and perform a variety of everyday tasks. The system is developed using the Python programming language and integrates several open-source libraries and APIs to enhance its capabilities.

The core functionality is built around voice interaction. Using the SpeechRecognition library, the assistant captures the user's speech through a microphone and converts it into text. Once the spoken input is recognized, the program processes the command and determines the appropriate response or action. The assistant can respond verbally using the pyttsx3 library, which converts text into human-like speech.

This virtual assistant can perform several useful tasks. It can announce the current time, play music on YouTube, answer questions by pulling summaries from Wikipedia, tell programming-related jokes via pyjokes, and even check live weather updates for any city using the OpenWeatherMap API. For all other queries, it can perform a Google search using the pywhatkit library.

2.3 Implementation

```python
import speech_recognition as sr
import pyttsx3
import datetime
import pywhatkit
import wikipedia
import pyjokes
import requests

listener = sr.Recognizer()
alexa = pyttsx3.init()
voices = alexa.getProperty('voices')
alexa.setProperty('voice', voices[1].id)

API_KEY = "8027d0de43f998604a2b721f052f2e0a"
CITY = "chittagong"

def talk(text):
    print("Alexa:", text)
    alexa.say(text)
    alexa.runAndWait()

def take_command():
    try:
        with sr.Microphone() as source:
            print('Listening...')
            voice = listener.listen(source)
            command = listener.recognize_google(voice)
            print("You said:", command)
            return command.lower()
    except Exception as e:
        print("Error:", e)
        talk("Sorry, I didn't catch that.")
        return ""

def get_weather():
    try:
        print("Fetching weather info...")
        url = f"https://api.openweathermap.org/data/2.5/
           weather?q={CITY}&appid={API_KEY}&units=metric"
        response = requests.get(url)
        data = response.json()

        if data["cod"] == 200:
            temp = round(data["main"]["temp"])
            description = data["weather"][0]["description"].
               capitalize()
            weather_report = f"The current temperature in {
               CITY} is {temp} degrees Celsius with {
               description}."
            print(weather_report)
            talk(weather_report)
        else:
            talk("Sorry, I couldn't find the weather
               information.")
    except Exception as e:
        print("Weather Error:", e)
        talk("Something went wrong while fetching the weather
           .")

def run_alexa():
    command = take_command()
    if command == "":
        return

    print("Command received:", command)

    if 'time' in command:
        time = datetime.datetime.now().strftime('%I:%M %p')
        print("Time:", time)
        talk('Current time is ' + time)

    elif 'play' in command:
        song = command.replace('play', '')
        talk('Playing ' + song)
        pywhatkit.playonyt(song)

    elif 'tell me about' in command:
        look_for = command.replace('tell me about', '')
        info = wikipedia.summary(look_for, 1)
        print(info)
        talk(info)

    elif 'joke' in command:
        joke = pyjokes.get_joke()
        print("Joke:", joke)
        talk(joke)

    elif 'date' in command:
        talk('Sorry bro, I am in another relation.')

    elif 'weather' in command or 'temperature' in command:
        get_weather()

    else:
        talk("I did not get that, but I will search it on
           Google.")
        pywhatkit.search(command)

while True:
    run_alexa()
```

Listing 2.1: Virtual assistant

2.3.1 Tools Technologies

- **Python:** Core programming language
- **SpeechRecognition:** Captures and interprets spoken voice commands
- **pyttsx3:** Converts text responses into speech (Text-to-Speech)
- **pywhatkit:** Plays YouTube videos and performs web searches
- **wikipedia:** Fetches short summaries of topics
- **pyjokes:** Provides random programming-related jokes
- **requests:** Sends HTTP requests to fetch weather data
- **OpenWeatherMap:** API Provides real-time weather information

Chapter 3

Performance Evaluation

3.1 Simulation Procedure

1. **Environment Setup:** Install all required libraries, set the API key, and ensure the microphone is working.

2. **Code Execution:** Run the Python script voice$_{a}ssistant.pythroughyourchosenIDEorterminal.$

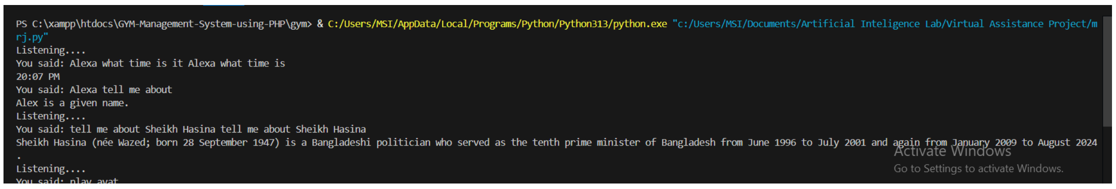
3. **Input:** Speak a valid command like "What time is it?", "Play Ayatul Qursi", "Tell me about Python", etc.

4. **Processing:** The assistant captures the audio input. Converts speech to text using Google's Speech API. Processes the command using conditional logic. Responds with the relevant output using pyttsx3 TTS.

5. **Output:** The assistant will: Speak the output aloud. Execute the task like open YouTube, fetch weather data, etc. Display relevant output in the terminal.

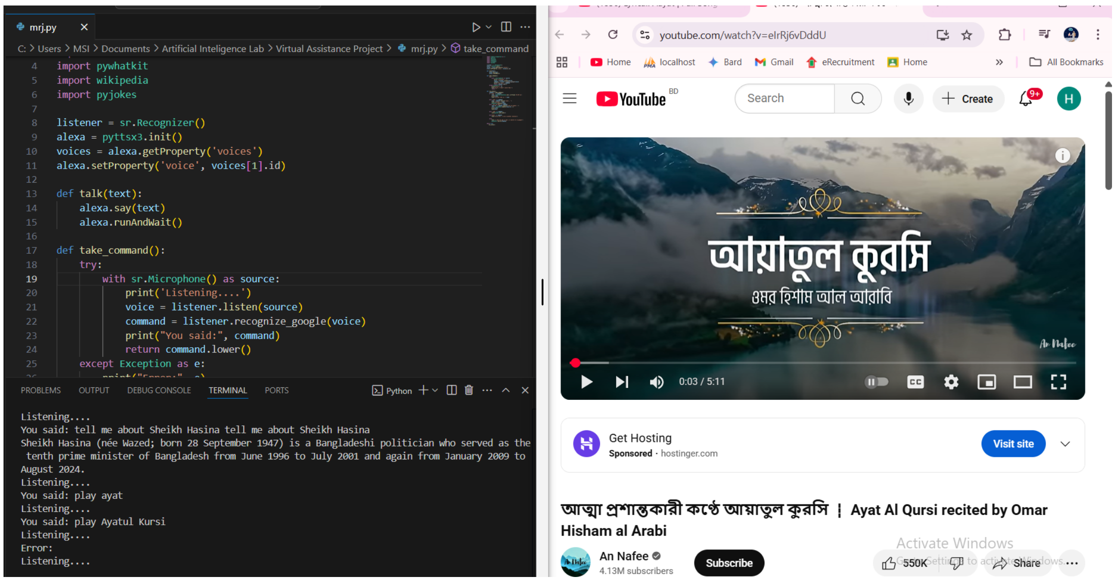
3.2 Results Analysis/Testing

[H]



```
PS C:\xampp\htdocs\GYM-Management-System-using-PHP\gym> & C:/Users/MSI/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/MSI/Documents/Artificial Inteligence Lab/Virtual Assistance Project/m
rj.py"
Listening....
You said: Alexa what time is it Alexa what time is
20:07 PM
You said: Alexa tell me about
Alex is a given name.
Listening....
You said: tell me about Sheikh Hasina tell me about Sheikh Hasina
Sheikh Hasina (née Wazed; born 28 September 1947) is a Bangladeshi politician who served as the tenth prime minister of Bangladesh from June 1996 to July 2001 and again from January 2009 to August 2024
.
Listening....
You said: play ayat
```

Figure 3.1: Telling time and searching wikipedia

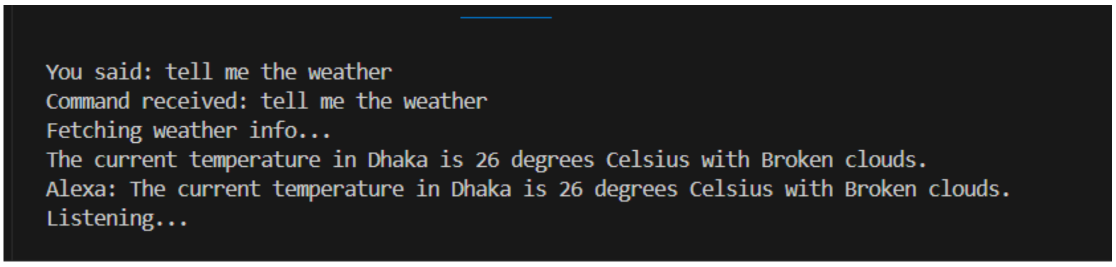

mrj.py
C: > Users > MSI > Documents > Artificial Inteligence Lab > Virtual Assistance Project > mrj.py > take_command
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
youtube.com/watch?v=elrRj6vDddU
Home localhost Bard Gmail eRecruitment Home All Bookmarks
YouTube BD
Search
Create
9+
H
আয়াতুল কুরসি
ওমর হিশাম আল আরাবি
An Nafee
0:03 / 5:11
Get Hosting
Sponsored · hostinger.com
Visit site
An Nafee
4.13M subscribers
Subscribe
550K
Share
Activate Windows
Go to Settings to activate Windows.

```
import pywhatkit
import wikipedia
import pyjokes

listener = sr.Recognizer()
alexa = pyttsx3.init()
voices = alexa.getProperty('voices')
alexa.setProperty('voice', voices[1].id)

def talk(text):
    alexa.say(text)
    alexa.runAndWait()

def take_command():
    try:
        with sr.Microphone() as source:
            print('Listening....')
            voice = listener.listen(source)
            command = listener.recognize_google(voice)
            print("You said:", command)
            return command.lower()
    except Exception as e:
```

```
Listening....
You said: tell me about Sheikh Hasina tell me about Sheikh Hasina
Sheikh Hasina (née Wazed; born 28 September 1947) is a Bangladeshi politician who served as the
 tenth prime minister of Bangladesh from June 1996 to July 2001 and again from January 2009 to
August 2024.
Listening....
You said: play ayat
Listening....
You said: play Ayatul Kursi
Listening....
Error:
Listening....
```

আত্মা প্রশান্তকারী কণ্ঠে আয়াতুল কুরসি ❘ Ayat Al Qursi recited by Omar Hisham al Arabi

Figure 3.2: Playing YouTube



```
You said: tell me the weather
Command received: tell me the weather
Fetching weather info...
The current temperature in Dhaka is 26 degrees Celsius with Broken clouds.
Alexa: The current temperature in Dhaka is 26 degrees Celsius with Broken clouds.
Listening...
```

Figure 3.3: Weather forecasting

3.3 Results Overall Discussion

The simulation of the Voice Assistant yielded successful results in performing a variety of basic tasks based on user voice commands. The assistant was able to understand and respond to several spoken inputs such as checking the time, playing songs on YouTube, retrieving Wikipedia summaries, telling jokes, and providing weather updates. These outcomes demonstrate the practical functionality of voice-controlled systems using Python and open-source libraries. The assistant processed commands by converting speech to text through the SpeechRecognition library and matched keywords in the recognized command to trigger specific actions. The results were presented both visually and audibly using the "pyttsx3" text-to-speech engine. The use of APIs such as "OpenWeatherMap" and Wikipedia allowed the assistant to access dynamic, real-time data.

Chapter 4

Conclusion

4.1 Discussion

This project successfully demonstrated the integration of speech recognition, text-to-speech, and external APIs like YouTube and OpenWeatherMap. It could perform tasks such as telling the time, playing music, retrieving Wikipedia summaries, and providing weather updates. However, some challenges were encountered, including occasional inaccuracies in speech recognition, limited natural language understanding, and installation issues with certain libraries. The assistant also ran in an infinite loop, which could be inefficient for real-world applications. Despite these limitations, the project met its core objectives, offering a solid foundation for building more advanced voice-controlled systems in the future. Future work could focus on improving natural language processing, error handling, and adding features like hotword detection.

4.2 Limitations

Despite achieving the main goals of creating a functional virtual assistant, several limitations were observed during the development and testing phases. These limitations not only impacted the performance of the system but also pointed to areas for potential improvement and refinement.

- **Speech Recognition Accuracy:** The system struggled with noisy environments and variations in accents, leading to misinterpretations of commands. The accuracy depended heavily on audio quality, and improving this would require more advanced recognition models or better training datasets.
- **Limited Natural Language Understanding:** The assistant could only process specific commands with exact phrasing, limiting flexibility. Enhancing its Natural Language Processing (NLP) capabilities would allow it to better understand conversational language and diverse user inputs.
- **Error Handling:** The system's error handling was basic, providing minimal feedback when tasks failed. A more robust approach, with clearer error messages, would improve user experience.

- **Continuous Listening and Efficiency:** The assistant used an infinite loop, consuming resources even when idle. Incorporating hotword detection would make it more efficient and user-friendly.

4.3 Scope of Future Work

- **Enhanced Natural Language Processing (NLP):** To improve the assistant's understanding of more complex and conversational commands, integrating advanced NLP techniques such as intent recognition and entity extraction could help. This would allow the assistant to handle a wider variety of phrases and understand user intent more accurately.
- **Improved Speech Recognition:** The accuracy of speech recognition could be enhanced by using more robust models, possibly incorporating machine learning techniques to train the system to better recognize diverse accents and noisy environments. Implementing models like Google's Speech-to-Text API with custom training could help with this.
- **Error Handling and User Feedback:** A more sophisticated error-handling system should be developed to provide detailed feedback to the user. For example, if a command fails or an API request cannot be processed, the assistant could suggest alternative actions or display helpful error messages.
- **Hotword Detection and Energy Efficiency:** Implementing hotword detection like "Hey Siri" or "Okay Google" would make the assistant more energy-efficient by only activating when needed. This would prevent the assistant from running in an infinite loop and consuming unnecessary resources.
- **Multi-platform Support:** The assistant could be expanded to support different platforms (e.g., mobile devices, smart speakers) and integrate with home automation systems (like IoT devices). This would make the assistant more versatile and usable in a broader range of environments.