BACHELOR OF COMPUTER SCIENCE

(BSc) PROGRAMME

CSC 317: ARTIFICIAL INTELLIGENCE AND APPLICATIONS

INDVIVUAL PROJECT TITLE

SEKA: A PERSONAL VOICE ASSISTANT

SUBMITTED BY

Isabella Mercy Abuor (P15/136964/2019)

1. Introduction

Problem Definition

As the world is moving towards computing utopia, with almost all tasks digitized, there has been a need of managing digital tasks easily and using voice Assistants applications like SIRI, OK google and the rest among many, make our lives easier and improves the Human-Computer interaction and also can benefit people who are visually impaired to do tasks within the computer without using braille computers. Well, the assistants receive external data (such as movement, voice, light, GPS readings, visually defined markers,etc.) via the hardware's sensors for further processing - and take it from there to function accordingly.

Objectives

The main aim of this project is to create a Voice Assistant that users can easily communicate with.

1. Create a working Voice Assistant using Python (Make sure it listens to User's command and execute it accordingly.)
2. To implement it on an application Using Tkinter
3. To test the above.

Justification

1. Enable a highly engaging user experience

Users are more engaged with voice assistants than with any other interface. Users can ask for everything they want by speaking normally to the applications, and they can do it while multitasking. It can also be used by the visually impaired people to perform the needed tasks.

2. Less Time Consuming

One Of the main advantages of Voice searches is their rapidity. In fact, voice is reputed to be four times faster than a written search.

3. Offer 24/7 support, without the costs

Customer satisfaction and retention are critical for any application or service, and keeping support people available around the world 24/7 may be pricey. Instead, voice assistants may take the user's inquiries and guide them through the problem, gathering information in the same way that a genuine customer service agent would, and guiding the user to the best solution. This type

of assistance can not only provide information for the user, but also direct them to the appropriate process and assist them in entering the data required to ensure the user's success.

Related Works

There already exist a number of desktop virtual assistants. A few examples of current virtual assistants available in market are discussed in this section along with the tasks they can provide.

SIRI from Apple

SIRI is personal assistant software that interfaces with the user through voice interface, recognizes commands and acts on them. It learns to adapt to user's speech and thus improves voice recognition over time. It also tries to converse with the user when it does not identify the user request.

Some Commands it takes

- Call someone from my contacts list
- Launch an application on my iPhone
- Send a text message to someone
- Set up a meeting on my calendar for 9am tomorrow
- Set an alarm for 5am tomorrow morning
- Play a specific song in my iTunes library
- Enter a new note

Cortana from Windows

Cortana is Microsoft's personal productivity assistant that helps you save time and focus attention on what matters most.

Here are some of the things you can do with Cortana in Windows:

- Calendar and schedule assistance. Cortana can help you manage your calendar. ...
- Meeting help. ...
- Make lists and set reminders and alarms. ...
- Open apps. ...
- Get definitions and quick answers. ...
- Get weather and news updates.

2. Methodology

The approach used in solving this problem was the Waterfall Model.

The waterfall model describes a highly structured process for creating a new project that flows linearly from gathering requirements to design, implementation, and testing — a process that is rooted in best practices from industries where design changes even early-on can be cost prohibitive, like manufacturing and construction.

This model values completeness and quality over speed hence why it is chosen for this project.

The Project Requirements.

The User should be able to speak and the system should able to recognize the speech.

The System should be able to listen and respond with Voice.

The System should be able to do the task the user said (required it to do)

The System should be able to run 24/7

Analysis

The requirements are analyzed and the problem fully researched and understood. It is seen that there is a need for better and a more user-friendly way in which humans interact with the computers and since the world is going online, it would make it easier and more reliable to have Virtual Personal Assistants that run 24/7 which can communicate with their users just like one would in a normal face to face conversation to make them easily achieve their daily digital tasks.

The necessary programming language is Python and Pycharm IDE for easier installation of the necessary packages.

Design

The design layouts to reliable communication is created by a programming Language Python. The GUI appearance is also designed in a simple manner to ensure familiarity which makes the app easy to use.

Implement

The technical implementation of the Software is realized and coding in the Software is done.

Testing

The whole system is tested to ensure functionality.

3. System Design

Block Diagram

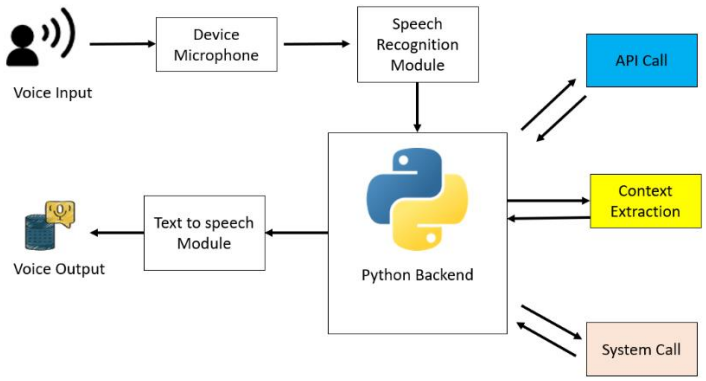

Figure 1:The Block Diagram of the System

Flowchart

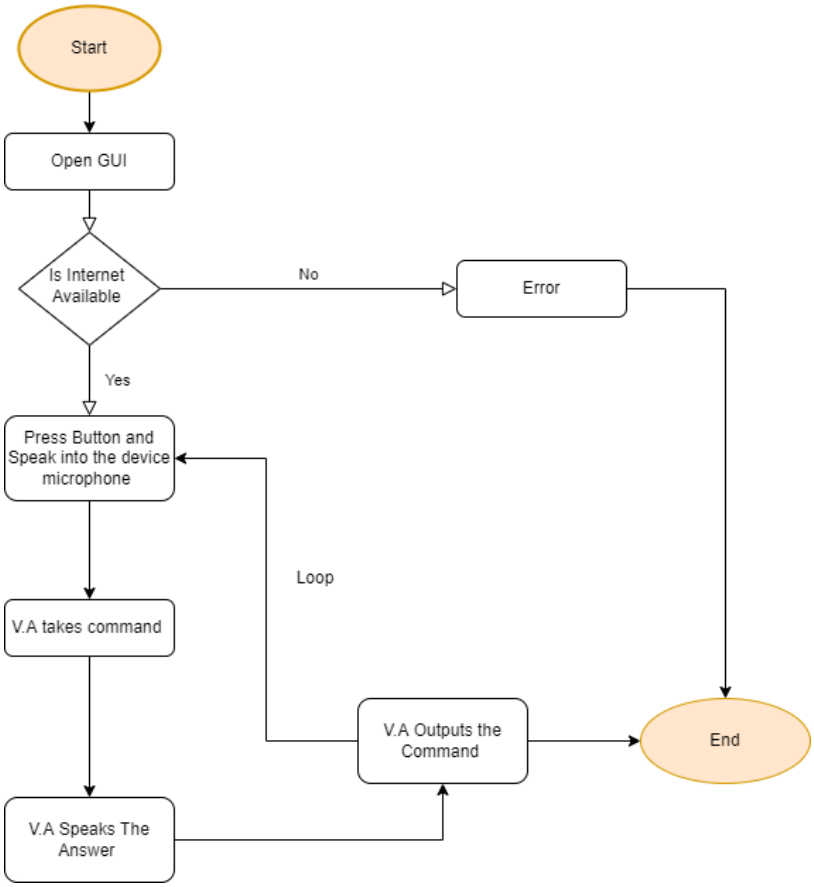

Figure 2: The System Flowchart

Implementation

In this section we are going to discuss in detail how the project was built from start to end.

First thing one must download Python and Pycharm IDE and Install the libraries in the terminal and import the following libraries in the code:

```
import speech_recognition as sr
import pyttsx3
import pywhatkit
import datetime
import wikipedia
import pyjokes
import os
from time import strftime
from tkinter import *
from PIL import Image, Image, ImageTk, ImageSequence
```

Figure 3:Importation of the needed libararies

Setting up the speech engine which the pyttsx3 module stores a variable name engine

```
listener = sr.Recognizer()
engine = pyttsx3.init()

voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)  # this [1] indicates a female voice
```

Figure 4: Code for setting the speech engine

Define a function **take_command** for the AI assistant to understand and to accept human language. The microphone captures the human speech and the recognizer recognizes the speech to give a response.

The exception handling is used to handle the exception during the run time error and, the **recognize_google** function uses google audio to recognize speech.

```
def take_command():
    try:
        with sr.Microphone() as source:
            talk('Hello,What can I do for you?')
            print('listening....')
            talk('listening....')
            voice = listener.listen(source)
            command = listener.recognize_google(voice)
            command = command.lower()
            if 'seka' in command:
                command = command.replace('seka', '')
                print(command)
    except:
        pass
    return command
```

Figure 5:Code for the Assistant to capture the Users voice

The main function starts from here, the commands given by the humans is stored in the variable **command**.

```
def run_seka():
    command = take_command()
    print(command)
```

Figure 6:Code to Store the Users command

If the following trigger words are there in the statement given by the users it invokes the virtual assistant to speak the below following commands.

```
elif 'hello' in command or 'hi' in command or 'hey' in command:
    day_time = int(strftime('%H'))
    if day_time < 12:
        talk("""
        Hello. Good Morning
        I am Seka and I am your personal voice assistant, Please give a command or say "help me"
        """)
```

Figure 7:Code for showing an example of a command

And the same goes for all the other commands and what the system can do for you for example play you a song on YouTube, open a desktop application, Tell the current time, Tell a joke, Wikipedia Searches and many more.

```
elif 'time' in command:
    time = datetime.datetime.now().strftime('%I:%M %p')
    print(time)
    talk('Current time is ' + time)
```

Figure 8:Code showing the time Command

```
elif 'who is' in command:
    person = command.replace('who is', '')
    info = wikipedia.summary(person, 1)
    print(info)
    talk(info)
```

Figure 9:Code showing the Wikipedia command