International Journal For Technological Research In Engineering
Volume 10, Issue 9, May-2023 *ISSN (Online): 2347 - 4718*

A RESEARCH PAPER ON VIRTUAL ASSISTANT

[1]Reena Rani, [2]Prof. Nisha
[1]Research Scholar, [2]Assistant Professor
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
PKG Group of Institutions
KURUKSHETRA UNIVERSITY, KURUKSHETRA

Abstract— Jarvis is a Python programming companion. Point it at a Python function, and it will execute it. As soon as you change your code and save it, Jarvis will detect this and run the function again. If an exception is thrown, it will be displayed in the error panel. If you include some debugging statements in your code, they will be displayed in the debug panel. Finally, if you use the OpenSceneGraph Python bindings, you can output an OSG tree to the Jarvis interface. This way, you can immediately see the new 3D scene that your code creates.

Jarvis was inspired by the work of Bret Victor, in particular his talk Inventing on Principle. The central idea is that the feedback loop in programming should be as short as possible, so that you can see the effects of your code changes immediately or almost immediately. Jarvis implements a (small) subset of these ideas.

Keywords: Python Programming

1. INTRODUCTION

A virtual assistant (VA) is a type of software program or application that uses artificial intelligence (AI) and natural language processing (NLP) to interact with users and perform a range of tasks. It can be thought of as a digital assistant that provides administrative, organizational, and even personal support to users.

A virtual assistant can perform a variety of tasks, including scheduling appointments, sending emails, setting reminders, answering questions, making phone calls, and even conducting research. It is designed to mimic human conversation and can understand and respond to voice or text commands. VAs can be accessed through various devices, such as smartphones, tablets, or smart speakers.

Virtual assistants use AI and machine learning algorithms to improve their responses over time. They can learn from previous interactions with users, allowing them to provide more accurate and personalized responses to users' requests. This makes them a valuable tool for businesses, entrepreneurs, and individuals who want to save time and increase productivity.

In summary, virtual assistants are digital assistants that use AI and natural language processing to interact with users and perform a range of tasks. They can be accessed through various devices and are designed to improve productivity and save time. With the advancements in AI and machine learning, virtual assistants are expected to become even more sophisticated in the future, making them an indispensable tool for businesses and individuals alike.

2. SYSTEM DESIGN

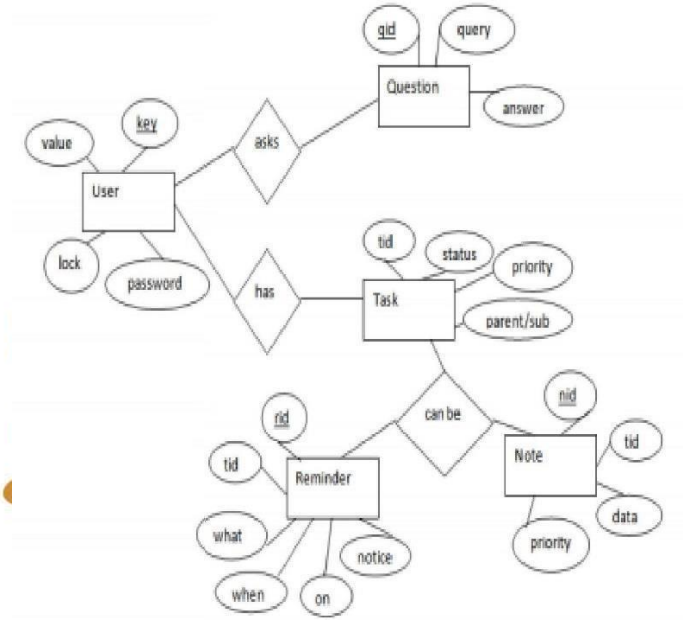
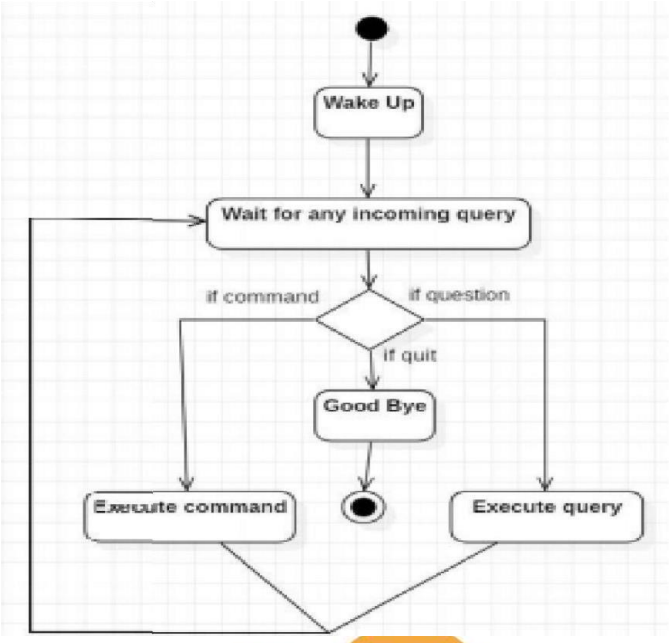


FIG 1. E-R DIAGRAM

The above diagram shows entities and their relationship for a virtual assistant system. We have a user of a system who can have their keys and values. It can be used to store any information about the user. Say, for key “name” value can be “Jim”. For some key’s user might like to keep secure. There he can enable lock and set a password (voice clip)

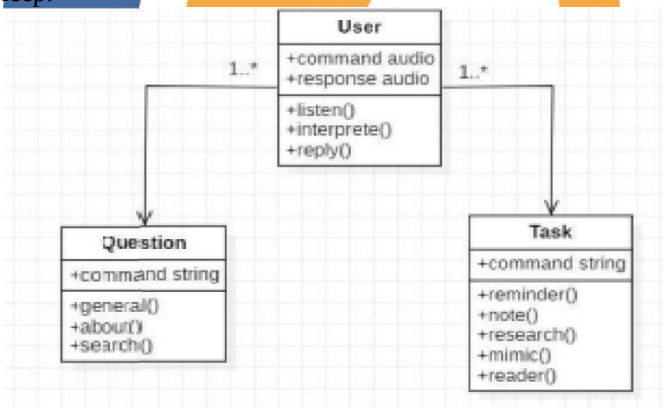
Single user can ask multiple questions. Each question will be given ID to get recognized along with the query and its corresponding answer.

User can also be having n number of tasks. These should have their own unique id and status i.e., their current state. A task should also have a priority value and its category whether it is a parent task or child task of an older task.



Activity Diagram

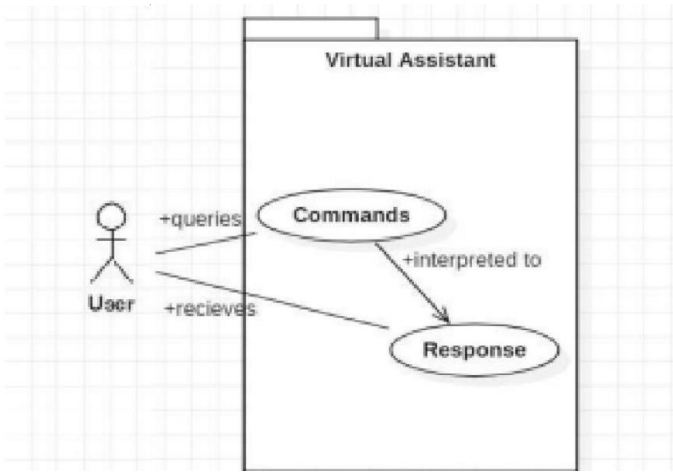
Initially, the system is in idle mode. As it receives any wake-up call it begins execution. The received command is identified whether it is a questionnaire or a task to be performed. Specific action is taken accordingly. After the Question is being answered or the task is being performed, the system waits for another command. This loop continues unless it receives quit command. At that moment, it goes back to sleep.



Class Diagram

The class user has 2 attributes command that it sends in audio and the response it receives which is also audio. It performs function to listen the user command. Interpret it and then reply sends back or response accordingly.

Question class has the command in string form as it is interpreted by interpret class. It sends it to general or about or search function based on its identification. The task class also has interpreted command in string format. It has various functions like reminder, note, mimic, research, and reader.



Use Case Diagram

In this project there is only one user. The user queries command to the system. System then interprets it and fetches answer. The response is sent back to the user.

3. Set Up JARVIS with Python

Before we start defining a few important functions, let's create a speech engine first. import pyttsx3 from decouple import config

```
USERNAME = config('USER') BOTNAME
=  config('BOTNAME')
engine = pyttsx3.init('sapi5')

# Set Rate engine.setProper ty('rate',190)
# Set Volume engine.setProperty('volume',1.0)
# Set Voice (Female) voices =
engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)
```

First, we have initialized an engine using the pyttsx3 module. sapi5 is a Microsoft Speech API that helps us use the voices.
Now, we can get the voices from the engine using the getProperty method. Voices will be a list of voices available in our system. If we print it, we can see as below:

```
[<pyttsx3.voice.Voice object at
0x000001AB9FB834F0>,
<pyttsx3.voice.Voice object at
0x000001AB9FB83490>]
```


4. HOW TO TAKE USER INPUT

We use this function to take the commands from the user and recognize the command using the speech recognition module.

```
import speech recognition as srfrom random import
choice from utils import opening text def
take_user_input():
"""Takes user input, recognizes it using Speech
Recognitionmodule and converts it into text"""

r = sr.Recognizer() with sr.Microphone() as source:
print('Listening ..... ')
r.pause_threshold = 1 audio =
r.listen(source)
try:
print('Recognizing .....') query =
r.recognize_google(audio, language='en-in')if not
'exit' in query or 'stop' in query:
speak(choice(opening_text)) else:
hour = datetime.now().hour if hour >= 21 and hour
< 6:
speak("Good night sir, take care!")else: speak('Have
a good day sir!')exit() except Exception:
speak('Sorry, I could not understand. Could you
pleasesay that again?')
query = 'None'
return query
```

RFERENCES

- www.stackoverflow.com
- www.pythonprogramming.net
- www.codecademy.com
- www.tutorialspoint.com
- www.google.co.in
- www.python.org
- Learning Python – harshit vashisth
- edureka!
- Codewithharry
- Designing Personal Assistant Software for Task Management using Semantic Web Technologies and Knowledge Databases, PurushothamBotla
- Python code for Artificial Intelligence: Foundations of Computational Agents, David L. Poole and Alan K. Mackwort

