MINI PROJECT REPORT

ON

JINI (VIRTUAL ASSISTANT)

MINI PROJECT LAB (KCS-554)

Submitted by-
Devansh Vats
Roll Number.1900970139004
Lakshya Kumar
Roll Number.1900970139005
Devansh Tiwari
Roll Number.1809713035

Under the supervision of
Ranjit Kumar

Department of Information Technology
Galgotias college of Engineering and Technology
Greater Noida
December 2020

ACKNOWLEDGEMENT

We want to give special thanks to our Mini Project coordinator Ranjit kumar for the timely advice and valuable guidance during designing and implementation of this project work.

We also want to express my sincere thanks and gratitude to Dr. Sanjeev Kumar Singh, Head of Department (HOD), Information Technology Department for providing me with the facilities and for all the encouragement and support.

Finally, We express my sincere thanks to all staff members in the department of Information Technology branch for all the support and cooperation

Devansh Vats	Lakshya kumar	DevanshTiwari
(1900970139004)	(1900970139004)	(1809713035)
5th Semester	5th Semester	5th Semester

TABLE OF CONTENT

TITLE	PAGE

INTRODUCTION

We are all well aware about Cortana, Siri, Google Assistant and many other virtual assistants which are designed to aid the tasks of users in Windows, Android and iOS platforms.We also developed one such type of virtual assistant named as “JINI” for windows platform.It will communicate to the user via text/voice option.

This Software aims at developing a personal assistant for Window based systems. The main purpose of the software is to perform the tasks of the user at certain commands, provided in either of the ways, speech or text. It will ease most of the work of the user as a complete task can be done on a single command. Jini draws its inspiration from Virtual assistants like Cortana for Windows and Siri for iOS. Users can interact with the assistant either through voice commands or keyboard input.

Currently, the project aims to provide the Window Users with a Virtual Assistant that would not only aid in their daily routine tasks like searching the web, extracting weather data, vocabulary help and many others but also help in automation of various activities.
As a personal assistant, Jini assists the end-user with day-to-day activities like general human conversation, searching queries in various search engines like Google, Bing or Yahoo, searching for videos, retrieving images, live weather conditions, word meanings, searching for medicine details and reminding the user about the scheduled events and tasks. The user

statements/commands are analysed with the help of NLP to give an optimal solution.

It is based on Natural language process technology(NLP) which is the subset of an artificial intelligence.it will first understand the query entered by the user,then analyze it
and generate response according to the query and give it back to the user in the form of text or voice.It will be a personal assistant of the user. It will do all kind of work on the system provided by the user.

REQUIREMENTS

- **DEVELOPER END**

- Language->Python
- IDE->Pycharm
- GUI->PYQT5
- LIBRARY->Pyttsx3,Speech Recognition,Sys,Os,Web Browser ,Datetime, Wikipedia,Pywhatkit , Pyjokes, Random

- **USER END**

- Os->Window 7 Or Above
- Processor->I3 Or Above

MODULE DESCRIPTION

- **PYQT5**

PyQt5 is a set of Python bindings for v5 of the Qt application framework from The Qt Company.

Qt is a set of C++ libraries and development tools that includes platform independent abstractions for graphical user interfaces, networking, threads, regular expressions, SQL databases, SVG, OpenGL, XML, user and application settings, positioning and location services, short range communications (NFC and Bluetooth), web browsing, 3D animation, charts, 3D data visualisation and interfacing with app stores. PyQt5 implements over 1000 of these classes as a set of Python modules.

PyQt5 supports the Windows, Linux, UNIX, Android, macOS and iOS platforms.

PyQt does not include a copy of Qt. You must obtain a correctly licensed copy of Qt yourself. However, binary wheels of the GPL version of PyQt5 are provided and these include a copy of the appropriate parts of the LGPL version of Qt.

PyQt5 also contains a number of utility programs.

- **pyuic5** corresponds to the Qt **uic** utility. It converts QtWidgets based GUIs created using Qt Designer to Python code.
- **pyrcc5** corresponds to the Qt **rcc** utility. It embeds arbitrary resources (eg. icons, images, translation files) described by a resource collection file in a Python module.
- **pylupdate5** corresponds to the Qt **lupdate** utility. It extracts all of the translatable strings from Python code and creates or updates .ts translation files. These are then used by Qt Linguist to manage the translation of those strings.

- ## SPEECH RECOGNITION

Speech recognition has its roots in research done at Bell Labs in the early 1950s. Early systems were limited to a single speaker and had limited vocabularies of about a dozen words. Modern speech recognition systems have come a long way since their ancient counterparts. They can recognize speech from multiple speakers and have enormous vocabularies in numerous languages.

The first component of speech recognition is, of course, speech. Speech must be converted from physical sound to an electrical signal with a microphone, and then to digital data with an analog-to-digital converter. Once digitized, several models can be used to transcribe the audio to text.

Most modern speech recognition systems rely on what is known as a Hidden Markov Model(HMM). This approach works on the assumption that a speech signal, when viewed on a short enough timescale (say, ten milliseconds), can be reasonably approximated as a stationary process that is, a process in which statistical properties do not change over time

- ## PYTTSX3

Pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.In addition, this module has been tested and is known to work on the following systems:

- SAPI5 on Windows XP, Windows Vista, and Windows 8, 8.1, 10
- NSSpeechSynthesizer on Mac OS X 10.5 (Leopard) and 10.6 (Snow Leopard)
- eSpeak on Ubuntu Desktop Edition 8.10 (Intrepid), 9.04 (Jaunty).

FEATURES AND APPLICATION

- **Conversational Maturity**

Beyond understanding and interacting conversationally, a great assistant has specific natural language processing (NLP) capabilities to understand the context of a conversation in multiple languages. It can also identify the intent of a question–what is needed– to provide an accurate first response, and also propose options to confirm or clarify intent. It have advanced conversational capabilities and can proactively seek out information, and can also ask clarifying questions, even if the conversation isn't linear.

- **Free to Explore**

It can reach, consume, and process vast amounts of data– both structured and unstructured–to surface insights from any source - to gather relevant data to solve customer issues quickly.

- **Omni-Capable**

It converses seamlessly across multiple digital channels and retains data and context for a seamless experience.In best cases, even passing that information to a live agent if needed.

- **Multi Language Support**

User can interact with assistant in tiple language like english ad hindi.Language can be easily changed by simply giving commond to it.

- ## Emotionally Intelligent

It can infer user personality traits and understand sentiment and tone during an interaction to deliver a personalized experience, or escalate to a live-agent when necessary.

- ## COMMANDS THAT CAN BE GIVEN BY THE USER

Introduction.

JINI : My name is JINI,how may I help you?

Converses, barely.

Talk to JINI : how are you
JINI: I am fine ! How are you

Talk to JINI : Tell me something about yourself
JINI : I am Jini an A I based virtual assistant and i can help you lot like a your close friend !

Talk to JINI : like what
JINI : I can open specific website for u,i can play music online,i can open appliction,i can tell you date and time and even entertain u by cracking some funnny joke

Rhythmbox: Play, Pause, Open.

Talk to JINI : Play music
JINI : On it!
Talk to JINI : Play music
JINI : On it!
Talk to JINI : Please open rhythmbox jarvis
JINI : Right away, sir

Tells time.

Talk to JINI : what time is it?
JINI: The time is 4 43 am

Suggests Googling for all unrecognized interrogative questions

Talk to JINI : What is IIT, Bombay?
JINI : Do you want me to google that for you?
Talk to JINI : yes
JINI : Right away, sir! Created new window in existing browser session.

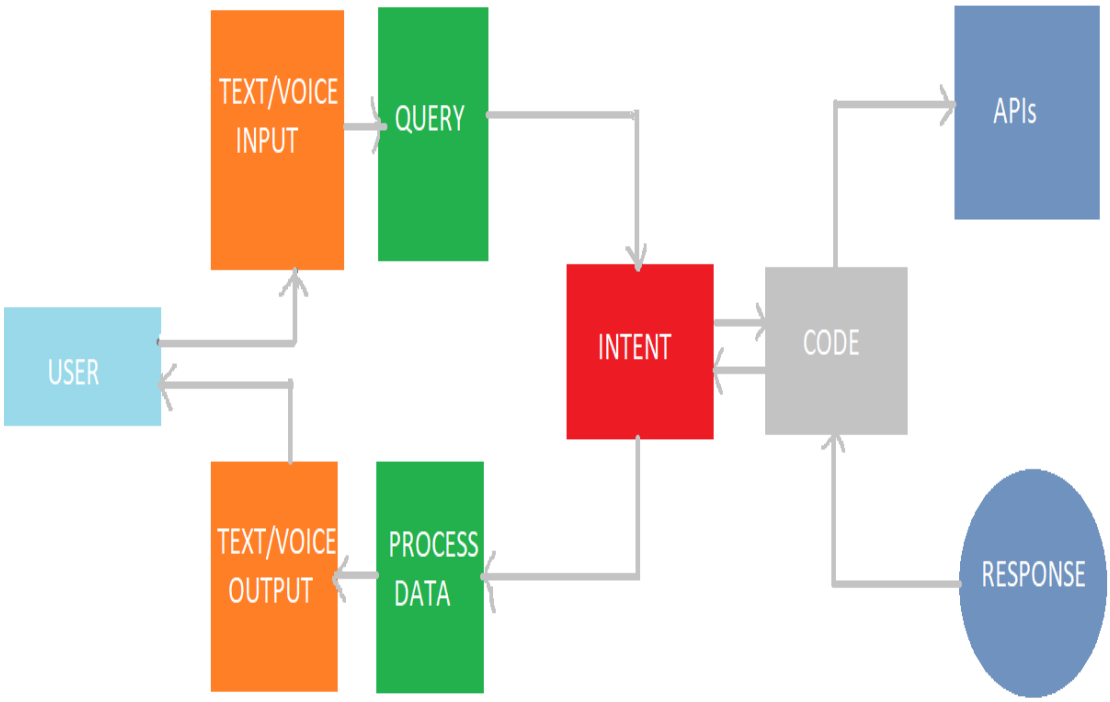
Plays any song, first search result in youtube

Talk to JINI : play me a song
JINI: What song, sir?
Talk to JINI : Alter Bridge Isolation
JINI : On it! Created new window in existing browser session.

Launches Programs.

Talk to JINI : open nautilus
JINI : Right away, sir!
Talk to JINI : take me to /etc
JINI : Sure thing! (Opens /etc in nautilus)
Talk to JINI : take me home
JINI : Sure thing! (Opens ~ in nautilus)
Talk to JINI: open chromium / open firefox / open calculator / open vlc
JINI: Sure thing!

DATA FLOW DIAGRAM

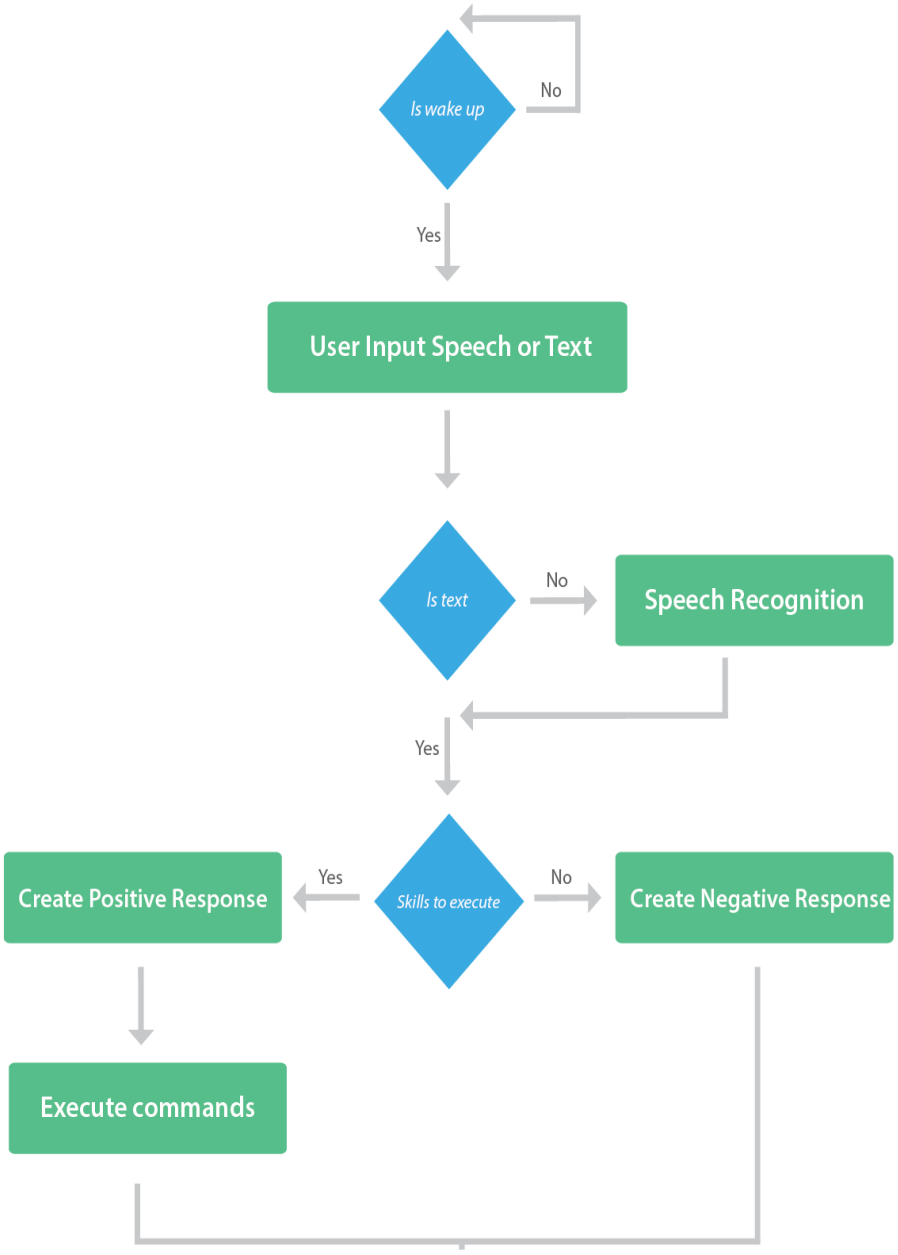


There are two level of data flow daigram in this virtual assistant 0 and 1.

0 Level:- In this level the query is given by the user in the form of text and speech which which will be understand and analyzed by the asssistant and generate the response accoding to the query.

1 Level:- What In this level the generated response is given back to the user in the form of text and speech by processing the data.

FLOW CHART


Is wake up
No
Yes
User Input Speech or Text
Is text
No
Speech Recognition
Yes
Skills to execute
Yes
Create Positive Response
No
Create Negative Response
Execute commands

USER CASE DIAGRAM

IMPLEMENTATION

```
from PyQt5 import QtWidgets,QtCore
from PyQt5.QtGui import QMovie
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.uic import loadUiType
import pyttsx3
import speech_recognition as sr
import sys
import os
import time
import webbrowser
import datetime
import wikipedia
import pywhatkit
import pyjokes
import random


flags = QtCore.Qt.WindowFlags(QtCore.Qt.FramelessWindowHint)
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice',voices[0].id)
engine.setProperty('rate',180)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def wishme():
    hour = int(datetime.datetime.now().hour)
    intro = "My name is jini,How may i help you"
    if hour>=0 and hour <12:
        speak("Good morning"+intro)
    elif hour>=12 and hour<18:
        speak("Good Afternoon"+intro)
    else:
        speak("Good night"+intro)

class mainT(QThread):
    def __init__(self):
        super(mainT,self).__init__()

    def run(self):
```

```
        self.JARVIS()

    def STT(self):
        R = sr.Recognizer()
        with sr.Microphone() as source:
            print("Listning.....")
            audio = R.listen(source)
        try:
            print("Recog.....")
            text = R.recognize_google(audio,language='en-in')
            print(">> ",text)
        except Exception:
            speak("Sorry Speak Again")
            return "None"
        text = text.lower()
        return text

    def JARVIS(self):
        wishme()
        while True:
            self.query = self.STT()
            if 'open google' in self.query:
                speak("opening...")
                chrome_path = 'C:/Program Files
(x86)/Google/Chrome/Application/chrome.exe %s'
                webbrowser.get(chrome_path).open("google.com")

            elif 'open youtube' in self.query:
                speak("opening...")
                chrome_path = 'C:/Program Files
(x86)/Google/Chrome/Application/chrome.exe %s'
                webbrowser.get(chrome_path).open("youtube.com")

            elif 'open a k t u' in self.query:
                speak("opening A.K.T.U website")
                chrome_path = 'C:/Program Files
(x86)/Google/Chrome/Application/chrome.exe %s'
                webbrowser.get(chrome_path).open("https://aktu.ac.in/")

            elif 'open stack overflow' in self.query:
                speak("opening stackoverflow...")
                chrome_path = 'C:/Program Files
(x86)/Google/Chrome/Application/chrome.exe %s'
                webbrowser.get(chrome_path).open("stackoverflow.com")

            elif 'open github' in self.query:
```

```
        webbrowser.open("https://www.github.com")
        speak("opening github")

    elif 'open facebook' in self.query:
        webbrowser.open("https://www.facebook.com")
        speak("opening facebook")

    elif 'open instagram' in self.query:
        webbrowser.open("https://www.instagram.com")
        speak("opening instagram")

    elif 'open google' in self.query:
        webbrowser.open("https://www.google.com")
        speak("opening google")

    elif 'open yahoo' in self.query:
        webbrowser.open("https://www.yahoo.com")
        speak("opening yahoo")

    elif 'open gmail' in self.query:
        webbrowser.open("https://mail.google.com")
        speak("opening google mail")

    elif 'open snapdeal' in self.query:
        webbrowser.open("https://www.snapdeal.com")
        speak("opening snapdeal")

    elif 'open amazon' in self.query or 'shop online' in self.query:
        webbrowser.open("https://www.amazon.com")
        speak("opening amazon")

    elif 'open flipkart' in self.query:
        webbrowser.open("https://www.flipkart.com")
        speak("opening flipkart")

    elif 'open ebay' in self.query:
        webbrowser.open("https://www.ebay.com")
        speak("opening ebay")

    elif 'who is' in self.query:
        speak("Searching...!")
        result = wikipedia.summary(self.query, sentences=2)
        print(result)
        speak("Acording To Wikipedia" + result)

    elif 'what is' in self.query:
```

```
            speak("Searching...!")
            result = wikipedia.summary(self.command, sentences=2)
            speak("Acording To Wikipedia" + result)

        elif 'play' in self.query:
            song = self.query.replace("play", "")
            speak("playing....!")
            pywhatkit.playonyt(song)

        elif "tell me the time" in self.query:
            time = datetime.datetime.now()
            speak("current time is" + time.strftime("%I" + "hour" + "%M"

        elif "tell me the date" in self.query:
            time = datetime.datetime.now()
            speak("today is" + time.strftime("%d" + "%B" + "%Y"))

        elif "tell me a joke" in self.query:
            speak(pyjokes.get_joke())

        elif "open sublime" in self.query:
            speak("opening...")
            path = "C:\\Program Files\\Sublime Text 3\\sublime_text.exe"
            os.startfile(path)
        elif "open sql" in self.query:
            speak("opening...")
            path = "C:\\Program Files\\MySQL\\MySQL Workbench 8.0
            os.startfile(path)
        elif "open powerpoint" in self.query:
            speak("opening...")
            path = "C:\\Program Files\\Microsoft
Office\\root\\Office16\\POWERPNT.EXE"
            os.startfile(path)

        elif "what\'s up" in self.query or 'how are you' in self.query:
            stMsgs = ['Just doing my thing!', 'I am fine!', 'Nice!', 'I am nice and full of
energy',
                      'i am okey ! How are you']
            ans_q = random.choice(stMsgs)
            speak(ans_q)

            """ans_take_from_user_how_are_you =takecomm()
            if 'fine' in ans_take_from_user_how_are_you or 'happy' in
ans_take_from_user_how_are_you or 'okey' in
ans_take_from_user_how_are_you:
                speak('okey..')
```

```
            elif 'not' in ans_take_from_user_how_are_you or 'sad' in
ans_take_from_user_how_are_you or 'upset' in
ans_take_from_user_how_are_you:
                speak('oh sorry..')"""

        elif 'who make you' in self.query or 'who created you' in self.query or 'who
develop you' in self.query:
            ans_m = " For your information Devansh Tiwari created me !"
            print(ans_m)
            speak(ans_m)

        elif "tell me something about yourself" in self.query or "who are you" in
self.query \
                or "your details" in self.query:
            about = "I am Jini an A I based virtual assistant and i can help you lot
like a your close friend !"
            print(about)
            speak(about)

        elif "like what" in self.query:
            help= "i can open specific website for u,i can play music online," \
                "i can open appliction,i can tell you date and time " \
                "and even entertain u by cracking some funnny joke "
            print(help)
            speak(help)

        elif "hello" in self.query or "hello Jarvis" in self.query:
            hel = "Hello Devansh Sir ! How May i Help you.."
            print(hel)
            speak(hel)

        elif "your name" in self.query or "sweet name" in self.query:
            na_me = "Thanks for Asking my name my self ! Jini"
            print(na_me)
            speak(na_me)

        elif "your feeling" in self.query:
            print("feeling Very good after meeting with you")
            speak("feeling Very good after meeting with you")

        elif self.query == 'none':
            continue

        elif "bye" in self.query:
            speak("ok good bye,have a nice day")
            exit()
```

```
FROM_MAIN,_ = loadUiType(os.path.join(os.path.dirname(__file__),"./scifi.ui"))

class Main(QMainWindow,FROM_MAIN):
    def __init__(self,parent=None):
        super(Main,self).__init__(parent)
        self.setupUi(self)
        self.setFixedSize(1920,1080)
        self.label_7 = QLabel
        self.exitB.setStyleSheet("background-image:url(./lib/exit - Copy.png);\n"
        "border:none;")
        self.exitB.clicked.connect(self.close)
        self.setWindowFlags(flags)
        Dspeak = mainT()
        self.label_7 = QMovie("./lib/gifloader.gif", QByteArray(), self)
        self.label_7.setCacheMode(QMovie.CacheAll)
        self.label_4.setMovie(self.label_7)
        self.label_7.start()

        self.ts = time.strftime("%A, %d %B")

        Dspeak.start()
        self.label.setPixmap(QPixmap("./lib/tuse.png"))
        self.label_5.setText("<font size=8 color='white'>"+self.ts+"</font>")
        self.label_5.setFont(QFont(QFont('Acens',8)))


app = QtWidgets.QApplication(sys.argv)
main = Main()
main.show()
exit(app.exec_())
```

SNAPSHOT

- **GUI INTERFACE**

CONCLUSION & FUTURE SCOPE

Through this voice assistant, we have automated various services using a single line command. It eases most of the tasks of the user like searching the web, retrieving weather forecast details, vocabulary help and general queries. We aim to make this project a complete server assistant and make it smart enough to act as a replacement for a general server administration. The future plans include integrating Jini with mobile using React Native to provide a synchronised experience between the two connected devices. Further, in the long run, Jini is planned to feature auto deployment supporting elastic beanstalk, backup files, and all operations which a general Server Administrator does. The functionality would be seamless enough to replace the Server Administrator with Jarvis.

REFRENCES

[1]Guide:NLP:-https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1

[2] Learning concepts of Machine Learning:-https://www.geeksforgeeks.org/introduction-machine-learning-using-python/#:~:text=Machine%20learning%20is%20a%20type,when%20exposed%20to%20new%20data.

[3]Tutorial help:- https://www.youtube.com/watch?v=Lp9Ftuq2sVI&t=2472s