**A Project report on**

**Desktop Assistance**

**A Dissertation submitted in partial fulfillment of the academic requirements for the award of the degree.**

**Bachelor of Technology**

**In**

**Computer Science & Engineering**

Submitted by

**K.Gayathri** (18H51A0546)

**M.Manasa** (18H51A05K6)

**T.Sai Bhavana** (18H51A05L6)

**Under the esteemed guidance of**

**Major Dr.V.A.Narayana**

**Principal**

****

**Department of Computer Science & Engineering**

**CMR College Of Engineering And Technology**

(An Autonomous Institution under UGC &JNTUH, Approved by AICTE, Permanently Affiliated to JNTUH, Accredited by NAAC with ’A+’ Grade.) KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

**2018- 2022**

pg. 1

Desktop assistant

**CMR COLLEGE OF ENGINEERING & TECHNOLOGY** KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

**CERTIFICATE**

This is to certify that the Mini Project-II report entitled **"Desktop Assistance"** being submitted by **, K.Gayathri** (18H51A0546),***.*M.Manasa *(***18H51A05K6),**T.Sai Bhavana** (18H51A05L6), in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering**is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree.

**Major Dr.V.A.Narayana Dr. K. Vijaya Kumar Principal Professor and HOD**

**CMRCET Dept. of CSE**

Submitted for viva voice Examination held on

**External Examiner**

CMRCET Page 2

Desktop assistant

**ACKNOWLEDGMENT**

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

I am grateful to **Major Dr.V.A.Narayana,** Principal CMRCET for his valuable suggestions and guidance during the execution of this project work.

I would like to thank **Dr. K. Vijaya Kumar**, Head of the Department of Computer Science and Engineering, for his moral support throughout the period of my study in CMRCET.

I am highly indebted to **Major Dr. V.A. Narayana**, Principal CMRCET for giving permission to carry out this project in a successful and fruitful way.

I would like to thank the Teaching & Non- teaching staff of Department of Computer Science and Engineering for their co-operation

Finally, I express my sincere thanks to **Mr. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

**KGayathri** (18H51A0546) **M.Manasa** (18H51A05K6) **T.Sai Bhavana** (18H51A05L6)

CMRCET Page 3

Desktop assistant

**TABLE OF CONTENTS**

| Chapter Number | Content | Page Number |
| --- | --- | --- |
|  |  |  |
| 1 | Introduction | 8 |
|  | 1.1 Present System | 8-9 |
|  | 1.2 Proposed System | 9-10 |
| 2 | System Design | 10 |
|  | 2.1 Data Flow | 11 |
| 3 | Software Details | 12 |
|  | 3.1 Visual Studio | 13 |
|  | 3.2 Python Libraries | 13-14 |
| 4 | Implementation Work Details | 15 |
|  | 4.1 Real Life Applications | 16 |
|  | 4.2 Data Implementation & Program Execution | 16-17 |
|  | 4.3 Functions | 17 |
| 5 | Source Code and Commands | 18-32 |
| 6 | Input/Output Screenshots | 33-38 |
| 7 | System Testing | 39 |
|  | 7.1 Functionality | 40 |
|  | 7.2 Usuability | 41 |
|  | 7.3 Security | 41 |
|  | 7.4 Stability | 41 |
| 9 | Conclusion | 42-43 |
|  |  |  |

CMRCET Page 4

Desktop assistant

LIST OF FIGURES

| Figure  Number | Figure Name | Page Number |
| --- | --- | --- |
|  |  |  |
| 2.1 | Data flow for STARK | 11 |
| 3.1 | Visual Studio IDE | 13 |
| 3.2 | Imported Modules | 14 |
| 6.1 | Input for google search | 34 |
| 6.2 | Output for google search | 34 |
| 6.3 | Input to send email | 34 |
| 6.4 | Output to send Email | 35 |
| 6.5 | Input for youtube search | 35 |
| 6.6 | Output for youtube search | 35 |
| 6.7 | Input to play music | 36 |
| 6.8 | Output to play music | 36 |
| 6.9 | Input to open cmd | 36 |
| 6.10 | Output to open cmd | 36 |
| 6.11 | Input & Output for Wikipedia search | 37 |
| 6.12 | Input to open Microsoft office | 37 |
| 6.13 | Output to open Microsoft office | 37 |
| 6.14 | Input to text message | 38 |
| 6.15 | Output to text message | 38 |
| 6.16 | Input and Output for Calculator | 38 |
| 7.1 | Input through voice commands | 40 |
| 7.2 | Output through voice commands | 40 |

CMRCET Page 5

Desktop assistant

**ABSTRACT**

**PROBLEM STATEMENT:**

To make an efficient voice assistant which performs actions based on the user command. **ABSTRACT:**

The Most famous application of windows is “CORTANA” which helps the end user to communicate end user laptop with voice and it also responds to the voice commands of the user. Same kind of application is also developed by the Google that is “Google Voice Search” which is used for in Android Phones. But this Application mostly works with Internet Connections. But our Proposed System has capability to work with and without Internet Connectivity. It is named as Personal Assistant with Voice Recognition Intelligence, which takes the user input inform of voice or text and process it and returns the output in various forms like action to be performed or the search result is dictated to the end user.

In addition, this proposed system can change the way of interactions between end user and the laptop. The system is being designed in such a way that all the services provided by the laptop are accessible by the end user on the user's voice commands. The project aims to develop a personal-assistant. Jarvis draws its inspiration from virtual assistants like Cortana for Windows. It has been designed to provide a user-friendly interface for carrying out a variety of tasks by employing certain well-defined commands.

CMRCET Page 6

Desktop assistant

**CHAPTER 1 INTRODUCTION**

CMRCET Page 7

Desktop assistant

Artificial Intelligence when used with machines, it shows us the capability of thinking like humans. In this, a computer system is designed in such a way that typically requires interaction from human. As we know Python is an emerging language so it becomes easy to write a script for Voice Assistant in Python. The instructions for the assistant can be handled as per the requirement of user. Speech recognition is the Alexa, Siri, etc. In Python there is an API called Speech Recognition which allows us to convert speech into text. It was an interesting task to make my own assistant. It became easier to send emails without typing any word, Searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. In the current scenario, advancement in technologies are such that they can perform any task with same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time.

As the voice assistant is using Artificial Intelligence hence the result that it is providing are highly accurate and efficient. The assistant can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. The assistant is no less than a human assistant but we can say that this is more effective and efficient to perform any task. The libraries and packages used to make this assistant focuses on the time complexities and reduces time.

The functionalities include , It can send emails, It can read PDF, It can send text on WhatsApp, It can open command prompt, your favorite IDE, notepad etc., It can play music, It can do Wikipedia searches for you, It can open websites like Google, YouTube, etc., in a web browser, It can give weather forecast, It can give desktop reminders of your choice. It can have some basic conversation.

Tools and technologies used are PyCharm IDE for making this project, and I created all py files in PyCharm. Along with this I used following modules and libraries in my project. pyttsx3, SpeechRecognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2, pyautogui, pyQt etc. I have created a live GUI for interacting with the JARVIS as it gives a design and interesting look while having the conversation.

**1.1 PRESENT SYSTEM**

We are familiar with many existing voice assistants like Alexa, Siri, Google Assistant, Cortana which uses concept of language processing, and voice recognition. They listens the command given by the user as per their requirements and performs that specific function in a very efficient and effective manner.

As these voice assistants are using Artificial Intelligence hence the result that they are providing are highly accurate and efficient. These assistants can help to reduce human effort and consumes time while

CMRCET Page 8

Desktop assistant

performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. These assistants are no less than a human assistant but we can say that they are more effective and efficient to perform any task. The algorithm used to make these assistant focuses on the time complexities and reduces time.

But for using these assistants one should have an account (like Google account for Google assistant, Microsoft account for Cortana) and can use it with internet connection only because these assistants are going to work with internet connectivity. They are integrated with many devices like, phones, laptops, and speakers etc.

**1.2 PROPOSED SYSTEM**

It was an interesting task to make my own assistant. It became easier to send emails without typing any word, Searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. Jarvis is different from other traditional voice assistants in terms that it is specific to desktop and user does not need to make account to use this.

The IDE used in this project is PyCharm. All the python files were created in PyCharm and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e. pyttsx3, SpeechRecognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2, pyautogui, pyQt etc. I have created a live GUI for interacting with the JARVIS as it gives a design and interesting look while having the conversation.

With the advancement Stark can perform any task with same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time. Functionalities of this project include, It can send emails, It can read PDF, It can send text on WhatsApp, It can open command prompt, your favorite IDE, notepad etc., It can play music, It can do Wikipedia searches for you, It can open websites like Google, YouTube, etc., in a web browser, It can give weather forecast.

CMRCET Page 9

Desktop assistant

**CHAPTER 2 SYSTEM DESIGN**

CMRCET Page 10

Desktop assistant

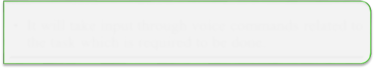
**2.1. DATA FLOW**

The data flow for Stark is as follow:

***Start***• Live GUI for interaction will appear on screen. • It will take input through voice commands related to 

***Input Perform*** 

***Exit***

******the task which is required to be done. 

• It will perform the required task for the user like opening notepad, searching on browser, sending mails, playing songs etc. 

• It keeps on asking for the command from user until the user say "Quit". Once the user say "Quit", it exits.

Figure 2.1 Data flow for STARK

The system is designed using the concept of Artificial Intelligence and with the help of necessary packages of Python. Python provides many libraries and packages to perform the tasks, for example pyPDF2 can be used to read PDF. The details of these packages are mentioned in Chapter 3 of this report. The data in this project is nothing but user input, whatever the user says, the assistant performs the task accordingly. The user input is nothing specific but the list of tasks which a user wants to get performed in human language i.e. English.

CMRCET Page 11

Desktop assistant

**CHAPTER 3**

**SOFTWARE DETAILS**

The IDE used in this project is Visual Studio.For this project following modules and libraries were used i.e. pyttsx3, SpeechRecognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2,

CMRCET Page 12

Desktop assistant

pyautogui, pyQt etc. I have created a live GUI for interacting with the JARVIS as it gives a design and interesting look while having the conversation.

**3.1. VISUAL STUDIO**

Visual Studio is an Integrated Development Environment(IDE) developed by Microsoft to develop GUI(Graphical User Interface), console, Web applications, web apps, mobile apps, cloud, and web services, etc. With the help of this IDE, you can create managed code as well as native code. It uses the various platforms of Microsoft software development software like Windows store, Microsoft Silverlight, and Windows API, etc. It is not a language-specific IDE as you can use this to write code in C#, C++, VB(Visual Basic), Python, JavaScript, and many more languages. It provides support for 36 different programming languages. It is available for Windows as well as for macOS.

**Figure 3.1 VisualStudio IDE**

**3.2. PYTHON LIBRARIES**

In JARVIS following python libraries were used:

**3.2.1. pyttsx3:** It is a python library which converts text to speech.

**3.2.2**. **SpeechRecognition:** It is a python module which converts speech to text. **3.2.3**. **pywhatkit:** It is python library to send WhatsApp message at a particular time with some additional features.

**3.2.4**. **Datetime:** This library provides us the actual date and time.

**3.2.5. Wikipedia:** It is a python module for searching anything on Wikipedia.

CMRCET Page 13

Desktop assistant

**3.2.6. Smtplib:** Simple mail transfer protocol that allows us to send mails and to route mails between mail servers.

**3.2.7. pyPDF2:** It is a python module which can read, split, merge any PDF.

**3.2.8. Pyjokes**: It is a python libararies which contains lots of interesting jokes in it. **3.2.9. Webbrowser:** It provides interface for displaying web-based documents to users. **3.2.10. Pyautogui:** It is a python libraries for graphical user interface.

**3.2.11. os:** It represents Operating System related functionality.

**3.2.12. sys:** It allows operating on the interpreter as it provides access to the variables and functions that usually interact strongly with the interpreter.



Figure 3.2 Imported Modules

CMRCET Page 14

Desktop assistant

**CHAPTER 4**

**IMPLEMENTATION WORK DETAILS**

STARK, a desktop assistant is a voice assistant that can perform many daily tasks of desktop like playing music, opening your favorite IDE with the help of a single voice command. Jarvis is different from other traditional voice assistants in terms that it is specific to desktop and user does not need to

CMRCET Page 15

Desktop assistant

make account to use this.

**4.1. REAL LIFE APPLICATION**

**4.1.1. Saves time:** STARK is a desktop voice assistant which works on the voice command offered to it, it can do voice searching, voice-activated device control and can let us complete a set of tasks. **4.1.2. Conversational interaction** It makes it easier to complete any task as it automatically do it by using the essential module or libraries of Python, in a conversational interaction way. Hence any user when instruct any task to it, they feel like giving task to a human assistant because of the conversational interaction for giving input and getting the desired output in the form of task done. **4.1.3. Reactive nature:** The desktop assistant is reactive which means it know human language very well and understand the context that is provided by the user and gives response in the same way, i.e. human understandable language, English. So user finds its reaction in an informed and smart way. **4.1.4. Multitasking:** The main application of it can be its multitasking ability. It can ask for continuous instruction one after other until the user “QUIT” it.

**4.1.5. No Trigger phase:** It asks for the instruction and listen the response that is given by user without needing any trigger phase and then only executes the task.

**4.2. DATA IMPLEMENTATION AND PROGRAM EXECUTION**

As the first step, install all the necessary packages and libraries. The command used to install the libraries is “*pip install*” and then import it. The necessary packages included are as follows: **4.2.1. LIBRARIES AND PACKAGES**

**4.2.2.1. pyttsx3:** It is a python library which converts text to speech.

**4.2.2.2**. **SpeechRecognition:** It is a python module which converts speech to text.

**4.2.2.3**. **pywhatkit:** It is python library to send WhatsApp message at a particular time with some additional features.

**4.2.2.4**. **Datetime:** This library provides us the actual date and time.

**4.2.2.5. Wikipedia:** It is a python module for searching anything on Wikipedia. **4.2.2.6. Smtplib:** Simple mail transfer protocol that allows us to send mails and to route mails between mail servers.

**4.2.2.7. pyPDF2:** It is a python module which can read, split, merge any PDF.

**4.2.2.8. Pyjokes**: It is a python libraries which contains lots of interesting jokes in it.

CMRCET Page 16

Desktop assistant

**4.2.2.9. Webbrowser:** It provides interface for displaying web-based documents to users. **4.2. 2.10. Pyautogui:** It is a python librariy for graphical user interface.

**4.2.2.11. os:** It represents Operating System related functionality.

**4.2.2.12. sys:** It allows operating on the interpreter as it provides access to the variables and functions that usually interact strongly with the interpreter.

**4.3 FUNCTIONS**

**4.3.1. takeCommand():** The function is used to take the command as input through microphone of user and returns the output as string.

**4.3.2. wishMe():** This function greets the user according to the time like Good Morning, Good Afternoon and Good Evening.

**4.3.3. taskExecution():** This is the function which contains all the necessary task execution definition like sendEmail(), pdf\_reader(), news() and many conditions in if condition like “open google”, “open notepad”, “search on Wikipedia” ,”play music” and “open command prompt” etc.

CMRCET Page 17

Desktop assistant

**CHAPTER 5**

**SOURCE CODE & COMMANDS**

import datetime

import os

import smtplib

import subprocess

CMRCET Page 18

Desktop assistant

import sys

import time

import webbrowser

import cv2

import psutil

import pyjokes

import pyttsx3

import pywhatkit as kit

import requests

import speech\_recognition as sr

import speedtest

import wikipedia

import winshell

import wolframalpha

from PyQt5 import QtCore, QtGui, QtWidgets from PyQt5.QtCore import \*

from PyQt5.QtCore import QDate, Qt, QTime, QTimer from PyQt5.QtGui import \*

from PyQt5.QtGui import QMovie

from PyQt5.QtWidgets import \*

from PyQt5.uic import loadUiType

from twilio.rest import Client

from coraUI import Ui\_MainWindow

engine = pyttsx3.init('sapi5')

engine.setProperty('rate', 140)

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[1].id)

def speak(audio):

engine.say(audio)

print(f"stark : {audio}")

engine.runAndWait()

def wishMe():

CMRCET Page 19

Desktop assistant

hour = int(datetime.datetime.now().hour)

if 0 <= hour < 12:

speak("Good Morning ")

elif 12 <= hour < 18:

speak("Good Afternoon ")

else:

speak("Good Evening ")

assname = "stark"

speak("I am your Assistant " + assname + " how can i help you")

def sendEmail(to, content):

server = smtplib.SMTP('smtp.gmail.com', 587)

server.ehlo()

server.starttls()

server.login('nikhilgattu9@gmail.com', '#Nikhilkumar2000')

server.sendmail('nikhilgattu9@gmail.com', to, content)

server.close()

def news():

main\_url = 'http://newsapi.org/v2/top headlines?sources=techcrunch&apikey=06416390f6324ef8af466e86cb99b4e5'

main\_page = requests.get(main\_url).json()

articles = main\_page["articles"]

head = []

day = ["first", "second", "third", "fourth", "fifth"]

for i in articles:

head.append(i["title"])

for i in range(len(day)):

speak(f"today's {day[i]} news is: {head[i]}")

def playgame(self):

speak("Welcome to Tic Tac Toe game!")

theBoard = {'7': ' ', '8': ' ', '9': ' ',

'4': ' ', '5': ' ', '6': ' ',

'1': ' ', '2': ' ', '3': ' '}

board\_keys = []

CMRCET Page 20

Desktop assistant

for key in theBoard:

board\_keys.append(key)

def printBoard(board):

print(" " +

board['7'] + '|' + board['8'] + '|' + board['9'])

print(" "+'-+-+-')

print(" " +

board['4'] + '|' + board['5'] + '|' + board['6'])

print(" "+'-+-+-')

print(" " +

board['1'] + '|' + board['2'] + '|' + board['3'])

def game():

turn = 'X'

count = 0

for i in range(10):

printBoard(theBoard)

print("It's your turn," + turn + " Move to which place?") speak("It's your turn " + turn + " Move to which place?")

move = input()

if move.isdigit():

if theBoard[move] == ' ':

theBoard[move] = turn

count += 1

else:

print("That place is already filled.")

speak("That place is already filled.")

continue

if count >= 5:

if theBoard['7'] == theBoard['8'] == theBoard['9'] != ' ': # across the top printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\* " + turn + " won. \*\*")

CMRCET Page 21

Desktop assistant

speak(turn + " won.")

break

# across the middle

elif theBoard['4'] == theBoard['5'] == theBoard['6'] != ' ': printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\* " + turn + " won. \*\*")

speak(turn + " won.")

break

# across the bottom

elif theBoard['1'] == theBoard['2'] == theBoard['3'] != ' ': printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\* " + turn + " won. \*\*")

speak(turn + " won.")

break

# down the left side

elif theBoard['1'] == theBoard['4'] == theBoard['7'] != ' ': printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\* " + turn + " won. \*\*")

speak(turn + " won.")

break

elif theBoard['2'] == theBoard['5'] == theBoard['8'] != ' ': # down the middle printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\* " + turn + " won. \*\*")

speak(turn + " won.")

break

# down the right side

elif theBoard['3'] == theBoard['6'] == theBoard['9'] != ' ': printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\* " + turn + " won. \*\*")

speak(turn + " won.")

break

elif theBoard['7'] == theBoard['5'] == theBoard['3'] != ' ': # diagonal printBoard(theBoard)

print("\nGame Over.\n")

CMRCET Page 22

Desktop assistant

print(" \*\* " + turn + " won. \*\*")

speak(turn + " won.")

break

elif theBoard['1'] == theBoard['5'] == theBoard['9'] != ' ': # diagonal printBoard(theBoard)

print("\nGame Over.\n")

print(" \*\* " + turn + " won. \*\*")

speak(turn + " won.")

break

if count == 9:

print("\nGame Over.\n")

print("It's a Tie!!")

if turn == 'X':

turn = 'O'

else:

turn = 'X'

else:

print("please enter a valid number")

speak("please enter a valid number")

restart = input("Do want to play Again?(y/n)")

if restart == "y" or restart == "Y":

for key in board\_keys:

theBoard[key] = " "

game()

game()

def get\_num(self):

dic = {"send it to arawind": 9381146833, "send it to sivaiah sir": 9505838400, "send it to nikhil": 8019997494}

speak('whom do you want to send the message ')

inpp = self.takecommand()

print(inpp)

if inpp in dic.keys():

a = dic[inpp]

return a

CMRCET Page 23

Desktop assistant

elif "no thanks" in inpp:

return "none"

else:

speak('there is no contact named ' + inpp) return get\_num(self)

# def search\_wikihow(query, m=10, lang="en"): # return list(wikiHow.search(query, m, lang))

class MainThread(QThread):

def \_init\_(self):

super(MainThread, self).\_init\_()

def takecommand(self):

r = sr.Recognizer()

with sr.Microphone() as source:

print('listening...')

r.pause\_threshold = 1

r.adjust\_for\_ambient\_noise(source)

audio = r.listen(source, phrase\_time\_limit=5) try:

print('Recognizing...')

query = r.recognize\_google(audio, language='en-in') print(f"Human: {query}")

except Exception:

return "none"

query = query.lower()

return query

def run(self):

# speak("please say wakeup to continue..") # while True:

# self.query = self.takecommand()

# if "wake" in self.query or "hello" in self.query: self.pertask()

def pertask(self):

CMRCET Page 24

Desktop assistant

wishMe()

while True:

self.query = self.takecommand()

if 'open notepad' in self.query:

npath = "C:\\Windows\\notepad.exe"

os.startfile(npath)

elif 'game' in self.query:

playgame(self)

elif 'close notepad' in self.query:

speak("closing notepad")

os.system("taskkill /f /im notepad.exe")

elif 'close command prompt' in self.query:

speak("closing CMD")

os.system("taskkill /f /im cmd.exe")

elif 'open command prompt' in self.query:

npath = "C:\\Windows\\System32\\cmd.exe"

os.startfile(npath)

elif 'play' in self.query:

song = self.query.replace('play', '')

speak('playing' + song)

print(song)

kit.playonyt(song)

elif 'hello' in self.query or 'hey' in self.query:

speak("hello there how are you!")

elif 'who made you' in self.query or 'who created you' in self.query: speak("I have been created by team stark.")

elif 'who are you' in self.query:

speak('iam stark, I can perform various task\'s') elif 'tell me a joke' in self.query or 'joke' in self.query: speak(pyjokes.get\_joke())

elif 'how are you' in self.query:

speak("I am fine, Thank you")

elif 'fine' in self.query or "good" in self.query:

speak("It's good to know that your fine")

elif 'wikipedia' in self.query:

qquery = self.query.replace("wikipedia", "")

print(qquery)

info = wikipedia.summary(qquery, sentences=1)

CMRCET Page 25

Desktop assistant

print(info)

speak('According to wikipedia ' + info) elif 'open youtube' in self.query:

speak('here you go')

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

speak('what should i have to search ')

cm = self.takecommand().lower()

webbrowser.open(f"{cm}")

elif 'send a message' in self.query:

timef = datetime.datetime.now().strftime('%I:%M %p') timee = datetime.datetime.now().strftime('%H:%M') s1 = str(timee[0]) + str(timee[1])

s2 = str(timee[3] + timee[4])

if s2[0] == '0':

s2 = s2[1]

elif 'AM' in timef and s1 == '12':

s1 = '00'

speak('what i have to send ')

mess = self.takecommand()

number = get\_num(self)

if number != "none":

speak('message sending..')

kit.sendwhatmsg('+91' + str(number), mess, int(s1), int(s2) + 2)

else:

pass

elif 'send a email' in self.query:

try:

speak("What should I say?")

content = self.takecommand().lower() to = input("enter email : ")

sendEmail(to, content)

speak("Email has been sent !")

except Exception as e:

print(e)

speak("I am not able to send this email") elif "write a note" in self.query:

CMRCET Page 26

Desktop assistant

speak("What should i write, ")

note = self.takecommand()

file = open('jarvis', 'w')

speak("Should i include date and time")

snfm = self.takecommand()

if 'yes' in snfm or 'sure' in snfm:

timee = datetime.datetime.now().strftime('%I:%M %p')

file.write(timee)

file.write(" :- ")

file.write(note)

speak('notes taken!')

else:

file.write(note)

speak('notes taken!')

elif "calculate" in self.query:

app\_id = "E4AJ9J-WGXTR9AGEX"

client = wolframalpha.Client(app\_id)

ind = self.query.lower().split().index("calculate")

qquery = self.query.split()[ind + 1:]

res = client.query(" ".join(qquery))

answer = next(res.results).text

speak("The answer is " + answer)

elif "stark what is" in self.query or "stark who is" in self.query or "birth" in self.query: try:

app\_id = "E4AJ9J-WGXTR9AGEX"

client = wolframalpha.Client(app\_id)

ind = self.query.lower().split().index("is")

qquery = self.query.split()[ind + 1:]

res = client.query(" ".join(qquery))

answer = next(res.results).text

speak(answer)

except Exception:

speak("unable to fetch the information")

elif 'show me the notes' in self.query:

speak("Showing Notes")

file = open("jarvis", "r")

s = file.read()

print(s)

CMRCET Page 27

Desktop assistant

speak(s)

elif "restart" in self.query:

subprocess.call(["shutdown", "/r"])

exit()

elif 'empty recycle bin' in self.query:

winshell.recycle\_bin().empty(confirm=False, show\_progress=True, sound=True) speak("Recycle Bin Recycled")

elif "hibernate" in self.query or "sleep" in self.query:

speak("Hibernating")

subprocess.call(["shutdown", "/h"])

exit()

elif "log off" in self.query or "shutdown" in self.query:

speak("Make sure all the application are closed before sign-out") time.sleep(10)

subprocess.call(["shutdown", "/l"])

exit()

elif 'no thanks' in self.query or 'thank you' in self.query:

speak('have a good day, thank you')

exit()

elif 'news today' in self.query or 'news' in self.query:

speak('please wait , fetching the latest news')

news()

elif "open camera" in self.query:

cap = cv2.VideoCapture(0)

while True:

ret, img = cap.read()

cv2.imshow('webcam', img)

k = cv2.waitKey(50)

if k == 27:

break

cap.release()

cv2.destroyAllWindows()

# elif "set alarm" in self.query:

# nn=int(datetime.datetime.now().hour)

# if nn==22:

# music\_dir=''

# songs=os.listdir(music\_dir)

# os.startfile(os.path.join(music\_dir,songs[0]))

CMRCET Page 28

Desktop assistant

elif 'send sms' in self.query:

speak('what should i have to send')

msg = self.takecommand()

account\_sid = 'AC1016d7c21e071e441350f7c9d774cd33' auth\_token = '9294e3235de29d9831295d5e93edcd9e' client = Client(account\_sid, auth\_token)

message = client.messages \

.create(

body=msg,

from\_='+14157797080',

to='+918019997494'

)

print(message.sid)

speak('message sent..')

elif 'where are we' in self.query:

speak('let me check!')

try:

ipadd = requests.get('https://api.ipify.org').text url = 'https://get.geojs.io/v1/ip/geo/' + ipadd + '.json' geo\_requests = requests.get(url)

geo\_data = geo\_requests.json()

city = geo\_data['city']

state = geo\_data['region']

country = geo\_data['country']

speak(

f" we are in state {state} of {city} city of {country} country") except Exception:

speak('unable to find location ')

pass

elif "weather" in self.query:

key = "4265b08b2aae7efa2e1c4456706c258d"

weather\_url = "http://api.openweathermap.org/data/2.5/weather?" ind = self.query.split().index("in")

location = self.query.split()[ind + 1:]

location = "".join(location)

url = weather\_url + "appid=" + key + "&q=" + location js = requests.get(url).json()

CMRCET Page 29

Desktop assistant

if js["cod"] != "404":

weather = js["main"]

temperature = weather["temp"]

temperature = temperature - 273.15

humidity = weather["humidity"]

weather\_response = " The temperature in Celcius is " + \ str(temperature) + " The humidity is " + str(humidity) speak(weather\_response)

else:

speak("City Not Found")

elif "where is" in self.query:

ind = self.query.lower().split().index("is")

location = self.query.split()[ind + 1:]

url = "https://www.google.com/maps/place/" + "".join(location) speak("This is where " + str(location[0]) + " is.") webbrowser.open(url)

# elif "activate how to do" in self.query:

# from pywikihow import search\_wikihow

# speak("activated how to do mod")

# while True:

# speak("tell me what you want to know")

# how = self.takecommand()

# try:

# if "exit" in how or "no thanks" in how: # speak("ok how to do mode is closed") # break

# else:

# m = 1

# h = search\_wikihow(how, m)

# assert len(h) == 1

# h[0].print()

# speak(h[0].summary)

# except Exception:

# speak("sorry something went wrong")

# break

elif "laptop percentage" in self.query:

battery = psutil.sensors\_battery()

per = battery.percent

speak(str(per)+" percent")

CMRCET Page 30

Desktop assistant

elif "internet speed" in self.query:

st = speedtest.Speedtest()

dl = st.download()

ul = st.upload()

speak("download speed is: "+str(dl) +

" upload speed is " + str(ul))

else:

print("\n")

startExecution = MainThread()

class Main(QMainWindow):

def \_init\_(self):

super().\_init\_()

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.ui.pushButton.clicked.connect(self.startTask) self.ui.pushButton\_2.clicked.connect(self.close)

def startTask(self):

self.ui.movie = QtGui.QMovie(

"../../Downloads/Iron man wallpaper - Imgur.gif") self.ui.label.setMovie(self.ui.movie)

self.ui.movie.start()

self.ui.movie = QtGui.QMovie(

"../../Downloads/Jarvis Loading Screen on Make a GIF.gif") self.ui.label\_2.setMovie(self.ui.movie)

self.ui.movie.start()

timer = QTimer(self)

timer.timeout.connect(self.showTime)

timer.start(1000)

startExecution.start()

def showTime(self):

current\_time = QTime.currentTime()

current\_date = QDate.currentDate()

label\_time = current\_time.toString('hh:mm:ss') label\_date = current\_date.toString(Qt.ISODate) self.ui.textBrowser.setText(label\_date)

CMRCET Page 31

Desktop assistant

self.ui.textBrowser\_2.setText(label\_time)

app = QApplication(sys.argv)

stark = Main()

stark.show()

exit(app.exec\_())

CMRCET Page 32

Desktop assistant

**CHAPTER 6**

**INPUT/OUTPUT SCREENSHOT**

****Figure 6.1 Input for Google search

CMRCET Page 33

Desktop assistant



Figure 6.2Output for Google search

Figure 6.3 Input to send Email

Figure 6.4 Output to send Email

CMRCET Page 34

Desktop assistant



Figure 6.5 Input for YouTube search

Figure 6.6 Output for YouTube search

 Figure 6.7 Input to play music

CMRCET Page 35

Desktop assistant



Figure 6.8 Output to play music

Figure 6.9 Input to open cmd

Figure 6.10 Output to open cmd

Figure 6.11 Input and output for Wikipedia search

CMRCET Page 36

Desktop assistant



Figure 6.12 Input to open Microsoft Office



Figure 6.13 Output to open Microsoft Office

 Figure 6.14.Input to text message

CMRCET Page 37

Desktop assistant



**Figure 6.15 Output to text message**

****Figure 6.16 Input and output for calculator

CMRCET Page 38

Desktop assistant

**CHAPTER 7**

**SYSTEM TESTING**

CMRCET Page 39

Desktop assistant

The system testing is done on fully integrated system to check whether the requirements are matching or not. The system testing for JARVIS desktop assistant focuses on the following four parameters:

**7.1. FUNCTIONALITY**

In this we check the functionality of the system whether the system performs the task which it was intended to do. To check the functionality each function was checked and run, if it is able to execute the required task correctly then the system passes in that particular functionality test. For example to check whether JARVIS can search on Google or not, as we can see in the figure 7.1, user said “Open Google”, then Jarvis asked, ”What should I search on Google?” then user said, “What is Python”, Stark open Google and searched for the required input.



**Figure 7.1 Input through voice commands**

**Figure 7.2 Output**

CMRCET Page 40

Desktop assistant

**7.2. USABILITY**

Usability of a system is checked by measuring the easiness of the software and how user friendly it is for the user to use, how it responses to each query that is being asked by the user. It makes it easier to complete any task as it automatically do it by using the essential module or libraries of Python, in a conversational interaction way. Hence any user when instruct any task to it, they feel like giving task to a human assistant because of the **conversational interaction** for giving input and getting the desired output in the form of task done.

The desktop assistant is **reactive** which means it know human language very well and understand the context that is provided by the user and gives response in the same way, i.e. human understandable language, English. So user finds its reaction in an informed and smart way.

The main application of it can be its **multitasking** ability. It can ask for continuous instruction one after other until the user “QUIT” it. It asks for the instruction and listen the response that is given by user without needing any **trigger phase** and then only executes the task.

**7.3. SECURITY**

The security testing mainly focuses on vulnerabilities and risks. As JARVIS is a local desktop application, hence there is no risk of data breaching through remote access. The software is dedicated to a specific system so when the user logs in, it will be activated.

**7.4. STABILITY**

Stability of a system depends upon the output of the system, if the output is bounded and specific to the bounded input then the system is said to be stable. If the system works on all the poles of functionality then it is stable.

CMRCET Page 41

Desktop assistant

**CHAPTER 8 CONCLUSION**

CMRCET Page 42

Desktop assistant

Stark is a very helpful voice assistant without any doubt as it saves time of the user by conversational interactions, its effectiveness and efficiency. But while working on this project, there were some limitations encountered and also realized some scope of enhancement in the future which are mentioned below:

**8.1. LIMITATIONS**

9.1.1. Security is somewhere an issue, there is no voice command encryption in this project. 9.1.2. Background voice can interfere

9.1.3. Misinterpretation because of accents and may cause inaccurate results.

9.1.4. JARVIS cannot be called externally anytime like other traditional assistants like Google Assistant can be called just by saying, “Ok Google!”

**8.2. SCOPE FOR FUTURE WORK**

9.2.1. Make Stark to learn more on its own and develop a new skill in it.

9.2.2. JARVIS android app can also be developed.

9.2.3. Make more Jarvis voice terminals.

9.2.4. Voice commands can be encrypted to maintain security.

CMRCET Page 43