

**Mini Project**  
(2020-2021)  
**Report on**  
**AI Jarvis Voice Assistant System**



Institute of Engineering & Technology

**Team Members:**

Suryansh Naugraiya (181500738)

Latesh Kumari (191599009)

Prathmesh Kumar Saini(181500497)

**Supervised By:**

Mr. Piyush Vashistha

(Assistant Professor)

Department of Computer Engineering & Applications

**GLA University**  
**Mathura-281406, India**



**Department of computer Engineering and Applications**

**GLA UNIVERSITY, MATHURA**

**17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,  
Mathura – 281406**

---

## **Declaration**

We hereby declare that the work which is being presented in the Mini Project “AI Jarvis Voice Assistant”, in partial fulfillment of the requirements for Mini Project viva voce, is an authentic record of our own work carried under the supervision of “Mr. Piyush Vashistha”.

**Signature of Candidate:**

**Name of Candidate:**

Suryansh Naugraiya (181500738)

Latesh Kumari (191599009)

Prathmesh Kumar Saini (181500497)

**Course: B.Tech. (CSE) Year: 3<sup>rd</sup> Semester: 6<sup>th</sup>**

## Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. On the completion of this project. We would like to extend our sincere thanks to all of them. We are highly indebted to this project guide **Mr. Piyush Vashistha Assistant Professor of Department of Computer Engineering and Applications of GLA University** for their guidance and constant supervision as well as for providing necessary information regarding the project. We wish to extend our sincere gratitude to **Prof. Anand Singh Jalal, Head of Department of Computer Engineering and Applications and faculty of CEA Department of GLA University** for their guidance, encouragement and give this opportunity and valuable suggestion which prove extremely useful and helpful in the completion of this report. We would also like to thank all those who directly or indirectly supported or helped us in completing our project in time. We would like to express our gratitude towards our parents and member of our college for their kind cooperation and encouragement which helped me in completion of this project. All of them have willingly helped us out with their abilities.

Thanks

Suryansh Naugraiya

Latesh Kumari

Prathmesh Kumar Saini

## **Abstract**

In the Modern Era of fast moving technology we can do things which we never thought we could do before but, to achieve and accomplish these thoughts there is a need for a platform which can automate all our tasks with ease and comfort. Thus we humans developed applications like Personal Voice Assistant having the ability to inter act with the surroundings just by one of the materialistic form of human interaction i.e .HUMAN VOICE.

The most famous application of android mobile phone is “Google Assistant”, “Google Voice Search” which is developed by the Google .Various applications like Microsoft Cortana, Amazon Alexa is also used as an voice assistant .The voice application of iphone is “SIRI” which helps the end user to communicate end-user mobile with voice and it also responds to the voice commands of the user. We are going to develop a web application were the voice assistant would be available for a particular desktop user. It can change the way of inter actions between user and the system. The Application is being designed in such a way that all the services provided by the system are accessible by the desktop user on the user's voice commands.

# Table of Content

Acknowledgement .....	iii
Abstract .....	iv
1.0 Introduction .....	1
2.0 About Desktop Voice Assistant .....	2
What is a Voice Assistant ?.....	2
Motivation .....	2
3.0 History of Voice Assistants .....	3
History of Voice Assistants .....	3
4.0 System Requirements Specification .....	4-6
Hardware Specification .....	4
Software Specification .....	4
Installation of Python and JupyterNotebook .....	5
Dependencies and requirements .....	6
5.0 System Implementation.....	7-12
6.0 Testing.....	13-15
7.0 Conclusion: What the future holds.....	16
8.0 References/Bibliography.....	17
<b>Appendices.....</b>	<b>17-30</b>

---

# 1.0 Introduction

---

## About the Project

As we know Python is a suitable language for script writers and developers. Let's write a script for Personal Voice Assistant using Python. The query for the assistant can be manipulated as per the user's need.

The implemented assistant can open up the application (if it's installed in the system), search Google, Wikipedia and YouTube about the query, calculate any mathematical question, etc by just giving the voice command. We can process the data as per the need or can add the functionality, depends upon how we code things.

We are using Google speech recognition API and google text to speech for voice input and output respectively.

Also, for calculating mathematical expression WolframAlpha API can be used.

Playsound Package is used to play the saved mp3 sound from the system.

## 2.0 About Desktop Voice Assistant

---

### What is a Voice Assistant?

A **voice assistant** or **intelligent personal assistant** is a software agent that can perform tasks or services for an individual based on verbal commands i.e. by interpreting human speech and respond via synthesized voices. Users can ask their assistants' questions, control home automation devices, and media playback via voice, and manage other basic tasks such as email, to-do lists, open or close any application etc with verbal commands.

Let me give you the example of Braina (Brain Artificial) which is an intelligent personal assistant, human language interface, automation and **voice recognition software** for Windows PC. Braina is a multi-functional AI software that allows you to interact with your computer using **voice commands** in most of the languages of the world. Braina also allows you to accurately convert speech to text in over 100 different languages of the world.

### Motivation

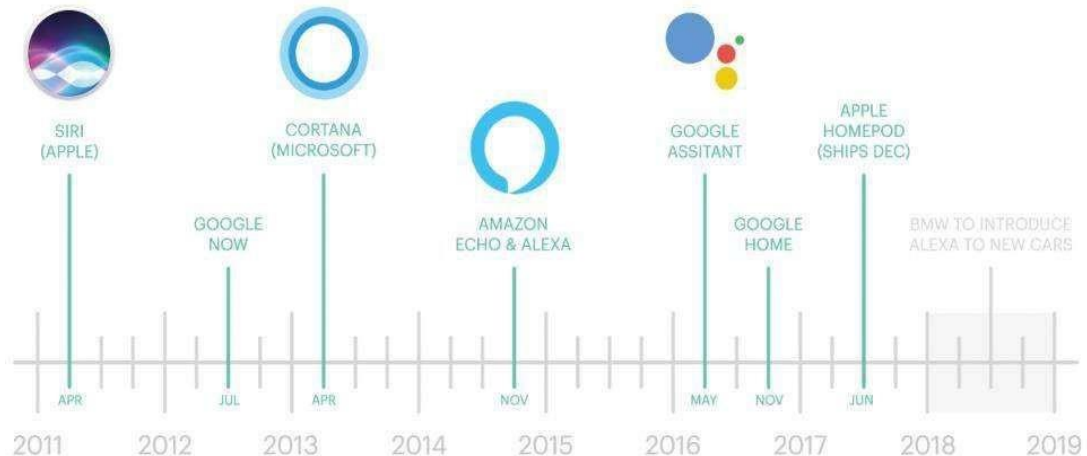
Who doesn't want to have the luxury to own an assistant who always listens for your call, anticipates your every need, and takes action when necessary? That luxury is now available thanks to artificial intelligence-based voice assistants.

Voice assistants come in somewhat small packages and can perform a variety of actions after hearing your command. They can turn on lights, answer questions, play music, place online orders and do all kinds of AI-based stuff.

Voice assistants are not to be confused with virtual assistants, which are people who work remotely and can, therefore, handle all kinds of tasks. Rather, voice assistants are technology based. As voice assistants become more robust, their utility in both the personal and business realms will grow as well.

## 3.0 History of Voice Assistants

### History of Voice Assistants



#### A modern history of Voice Assistants

In recent times, Voice assistants got the major platform after Apple integrated the most astonishing Virtual Assistant — Siri which is officially a part of Apple Inc. But the timeline of greatest evolution began with the year 1962 event at the Seattle World Fair where IBM displayed a unique apparatus called Shoebox. It was the actual size of a shoebox and could perform scientific functions and can perceive 16 words and also speak them in the human recognizable voice with 0 to 9 numerical digits.

During the period of the 1970s, researchers at Carnegie Mellon University in Pittsburgh, Pennsylvania — with the considerable help of the U.S Department of Defence and its Defence Advanced Research Projects Agency (DARPA) — made Harpy. It could understand almost 1,000 words, which is approximately the vocabulary of a three-year-old child.

Big organizations like Apple and IBM sooner in the 90s started to make things that utilized voice acknowledgment. In 1993, Macintosh began to building speech recognition with its Macintosh PCs with Plain Talk.



## 4.0 System Requirements Specification

### Hardware Requirements Specification

Processor: Intel Pentium III or later

Main Memory (RAM) : 4 GB

Cache Memory: 512 KB

Keyboard: 108 Keys

### Software Requirements Specification

Technology: Python 3, Different Libraries

Platform: Jupyter Notebook

Operating System: Windows 7, 8, 9, 10, XP

### Installation of Python





## Installation of Jupyter Notebook

For new users, we highly recommend installing Anaconda. Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

1. Download Anaconda. We recommend downloading Anaconda's latest Python 3 version (currently Python 3.7).
2. Install the version of Anaconda which you downloaded, following the instructions on the download page.
3. Congratulations, you have installed Jupyter Notebook. To run the notebook:

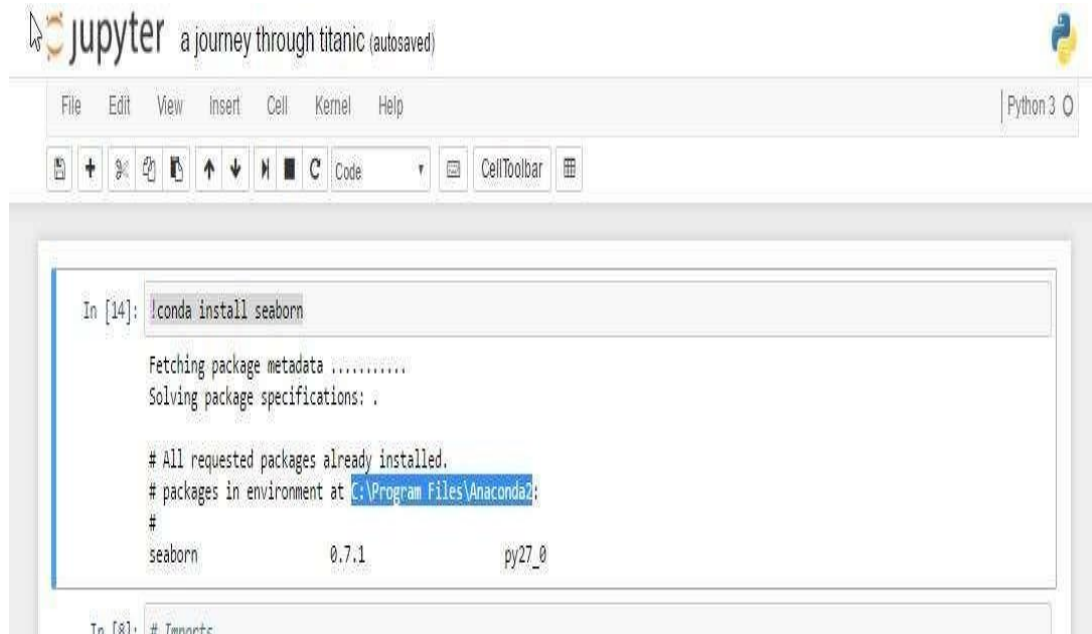
As an existing Python user, you may wish to install Jupyter using Python's package manager, pip, instead of Anaconda.

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

`pip3 install --upgrade pip`

Then install the Jupyter Notebook using:

`pip3 install jupyter`



## Dependencies and requirements:

Install all these python libraries :

`pip install SpeechRecognition`

`pip install beautifulsoup4 pip`

`install vlc pip install youtube-`

`dl pip install pyowm pip`

`install wikipedia`

## 5.0 System Implementation

---

### 1. Search anything in wikipedia.

The user can search anything on wikipedia just by saying “anything wikipedia” or “wikipedia anything”. For example : “shah rukh khan wikipedia” will search for shah rukh khan and tells you the first two line about it.

**How it works** : If you have said the phrase “anything wikipedia” or “wikipedia anything” in your command then it will replace the word “wikipedia” with the empty space and search for the word “anything” on wikipedia and tells you the first 2 lines about it.

### 2. Open any website in the browser.

You can open any website just by saying “open website\_name”.

For example: “Please open facebook” or “Hey, can you open google” like this you can ask Jarvis to open any website for you.

**How it works** : If you have said the word open in your command then it will search for website name in the user command using `re.search()`. Next, it will append the website name to `https://www.` and using web browser module the complete URL gets opened in the browser.

### 3. Send Email.

You can also ask your desktop assistant to send the email.

**How it works :** If you have said the word email in your command then the bot will ask for receipient, If my response is rajat, the bot will use pthons smtplib library.

The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon. Sending mail is done with Python's smtplib using an SMTP server. First it will intiate gmail SMTP using smtplib. SMTP(), then identify the server using ehlo() function, then encypting the session starttls(), then login to your mailbox using login(), then sending the message using sendmail().

#### **4. Launch any system application.**

Say "launch notepad" or "can you please launch vs code" etc, and Jarvis will launch that system application for you.

**How it works :** If you have said the word launch in your command then it will search for application name(if it is present in your system) in the user command using re.search(). It will then append the suffix ".app" to the application name. Now your application name is for example say calender.app(In macOS the executable files end with extension .app unlike in Windows which ends with .exe). So the executable application name will be launched using python subprocess's Popen() function. The subprocess module enables you to start new applications from your Python program.

#### **5. Tells you the current time.**

"Jarvis can you tell me the current time ?" or "what is the time now ?" and Jarvis will tell you the current time of your timezone.

**How it works :** If the keyword "time" is in your query then it will search for the current time and tell you.

## 6. Greetings/ leave

Say “ **hello Jarvis**” to greet your voice assistant or when you want the program to terminate say something like “**shutdown Jarvis**” or “**Jarvis please shutdown**” or “**bye bye jarvis**” or “**quit jarvis**” or “**bye jarvis**” or “**leave jarvis**” etc.

**How it works :** If you have said the word **hello** in your command, then depending on the time of the day, the bot will greet the user. If the time is more than 12 noon, the bot will respond “Hello Sir. Good afternoon”, likewise if the time is more than 6 pm, the bot will respond “Hello Sir. Good evening”. And when you give command as shutdown, `sys.exit()` will be called to terminate the program.

## 7. Tells you latest news feeds

Jarvis can also tell you the latest news update. The user just has to say “Jarvis what are the top news for today?” or “tell me the news for today”.

**How it works :** If you have said the phrase “news for today” in your command then it will scrape data using Beautiful Soup from Google News RSS() and read it for you. For convenience I have set number of news limit to 2.

## 8. Tells you the current weather and temperature of almost any city

Jarvis can also tell you the weather and temperature of any city around the world. The user just needs to say something like “what is the current weather in London” or “tell me the current weather in Delhi”.

**How it works :** If you have said the phrase “current weather” in your command then it will search for city name using `re.search()`. I have used python's `pyowm` library to get the weather of any city. `get_status()` will tell you about the weather condition like haze, cloudy, rainy etc and `get_temperature()` will tell you about the max and min temperature of the city.

## **9. Play you a song on VLC media player**

This feature allows your voice bot to play your desired song in VLC media player. The user will say “jarvis play me a song” or “play music” or “refresh me” or “do anything” the jarvis will randomly play a song from the list of saved song.

## **10. Google anything you ask.**

Your bot can fetch details of almost anything you ask her. Like “jarvis google about Salman khan” or “google semiconductors”. So as you can see you can ask about almost anything.

**How it works :** If you have said the phrase “google anything” in your command then it will search for the keyword in the user command using `re.search()`. Using python's wikipedia library it will search for that topic and extract first 500 characters(if you don't specify the limit the bot will read the whole page for you). Wikipedia is a Python library that makes it easy to access and parse data from Wikipedia.

## **11. Tells the stock price of AMAZON**

When you say that “jarvis tell the current stock price” then it searches the current stock price of amazon and tells you.

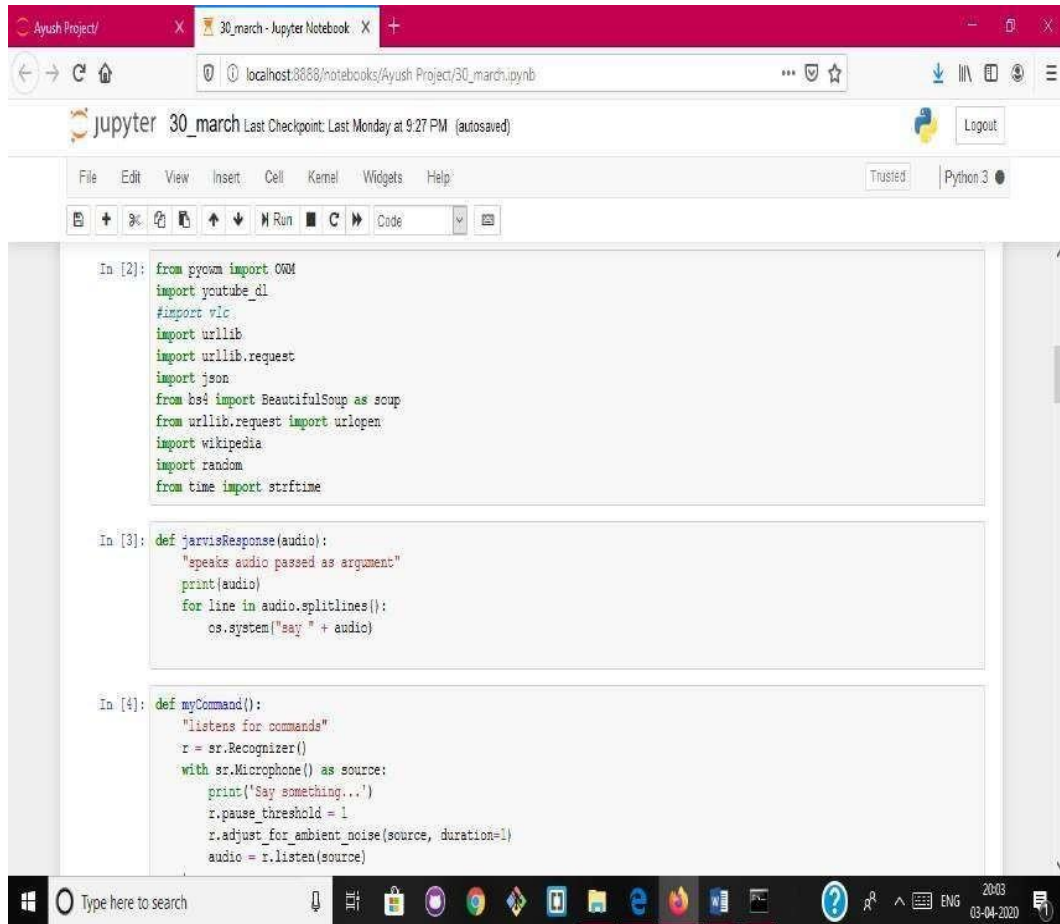
## **12. Tells the joke**

When you say that “jarvis tell me a joke” then it searches a joke from internet and tells you.

## **13. Tells you the city or state location**

When you say that “jarvis please tell the location of city/state” then it will search the location of that city or state and tells you.

## Execution



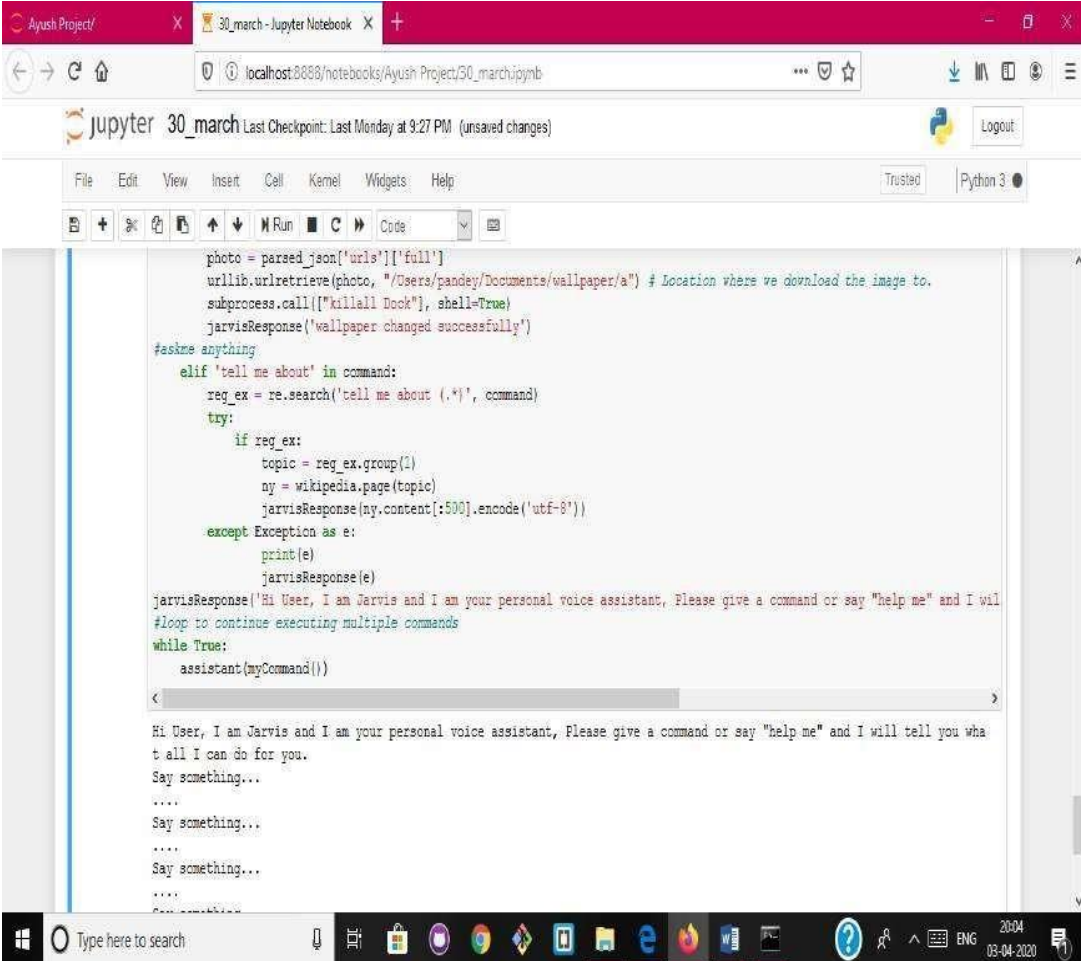
The screenshot displays a Jupyter Notebook titled '30\_march - Jupyter Notebook' running on a local host. The notebook contains three input cells with Python code. The first cell imports various libraries including gdown, youtube\_dl, vlc, urllib, requests, json, BeautifulSoup, urlopen, wikipedia, random, and time. The second cell defines a function 'jarvisResponse(audio)' that prints the audio and uses 'os.system' to play it. The third cell defines a function 'myCommand()' that listens for commands using 'sr.Recognizer' and 'sr.Microphone'.

```
In [2]: from gdown import GDM
import youtube_dl
#import vlc
import urllib
import urllib.request
import json
from bs4 import BeautifulSoup as soup
from urllib.request import urlopen
import wikipedia
import random
from time import strftime

In [3]: def jarvisResponse(audio):
    "speaks audio passed as argument"
    print(audio)
    for line in audio.splitlines():
        os.system("say " + audio)

In [4]: def myCommand():
    "listens for commands"
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print('Say something...')
        r.pause_threshold = 1
        r.adjust_for_ambient_noise(source, duration=1)
        audio = r.listen(source)
```





The screenshot displays a Jupyter Notebook environment with the following code and output:

```
photo = parsed_json['urls'][0]['full']
urllib.urlretrieve(photo, "/Users/pandey/Documents/wallpaper/a") # Location where we download the image to.
subprocess.call(["killall Dock"], shell=True)
jarvisResponse('wallpaper changed successfully')
#askme anything
elif 'tell me about' in command:
    reg_ex = re.search('tell me about (.*)', command)
    try:
        if reg_ex:
            topic = reg_ex.group(1)
            ny = wikipedia.page(topic)
            jarvisResponse(ny.content[:500].encode('utf-8'))
    except Exception as e:
        print(e)
        jarvisResponse(e)
jarvisResponse('Hi User, I am Jarvis and I am your personal voice assistant, Please give a command or say "help me" and I will
#loop to continue executing multiple commands
while True:
    assistant(myCommand())
```

Output:

```
Hi User, I am Jarvis and I am your personal voice assistant, Please give a command or say "help me" and I will tell you what
I can do for you.
Say something...
....
Say something...
....
Say something...
....
Say something...
```

## 6.0 Testing

---

The implementation phase of software development is concerned with translating design specification into source code. The preliminary goal of implementation is to write source code and internal documentation so that conformance of the code to its specifications can be easily verified, and so that debugging, testing and modifications are eased. This goal can be achieved by making the source code as clear and straightforward as possible. Simplicity, clarity and elegance are the hallmark of good programs, obscurity, cleverness, and complexity are indications of inadequate design and misdirected thinking.

Source code clarity is enhanced by structured coding techniques, by good coding style, by, appropriate supporting documents, by good internal comments, and by feature provided in modern programming languages.

The implementation team should be provided with a well-defined set of software requirement, an architectural design specification, and a detailed design description.

Each team member must understand the objectives of implementation. **Terms**

### **in Testing Fundamental**

#### **Error**

The term error is used in two ways. It refers to the difference between the actual output of software and the correct output, in this interpretation, error is essential a measure of the difference between actual and ideal. Error is also to used to refer to human action that result in software containing a defect or fault.

#### **Fault**

Fault is a condition that causes to fail in performing its required function. A fault is a basic reason for software malfunction and is synonymous with the commonly used term Bug.

## **Failure**

Failure is the inability of a system or component to perform a required function according to its specifications.

### **a. Unit Testing**

The term unit testing comprises the sets of tests performed by an individual programmer prior to integration of the unit into a larger system.

A program unit is usually small enough that the programmer who developed it can test it in great detail, and certainly in greater detail than will be possible when the unit is integrated into an evolving software product. In the unit testing the programs are tested separately, independent of each other. Since the check is done at the program level, it is also called program teasing.

### **b. Module Testing**

A module and encapsulates related component. So can be tested without other system module.

There are four categories of tests that a programmer will typically perform on a program unit.

**i** Functional test **ii**

Performance test **iii**

Stress test **iv**

Structure test

### **Functional Test**

Functional test cases involve exercising the code with Nominal input values for which expected results are known; as well as boundary values (minimum values, maximum values and values on and just outside the functional boundaries) and special values.

### **Performance Test**

Performance testing determines the amount of execution time spent in various parts of the unit, program throughput, response time, and device utilization by the program unit.

**Stress Test**

Stress test are those designed to intentionally break the unit. A great deal can be learned about the strengths and limitations of a program by examining the manner in which a program unit breaks.

---

**Structure Test**

Structure tests are concerned with exercising the internal logic of a program and traversing particular execution paths. Some authors refer collectively to functional performance and stress testing as “black box” testing. While structure testing is referred to as “white box” or “glass box” testing. The major activities in structural testing are deciding which path to exercise, deriving test data to exercise those paths, determining the test coverage criterion to be used, executing the test, and measuring the test coverage achieved when the test cases are exercised.

## 7.0 Conclusion: What the future holds

---

Throughout the history of computing, user interfaces have become progressively natural to use. The screen and keyboard were one step in this direction. The mouse and graphical user interface were another. Touch screens are the most recent development. The next step will most likely consist of a mix of augmented reality, gestures and voice commands. After all, it is often easier to ask a question or have a conversation than it is to type something or enter multiple details in an online form.

The more a person interacts with voice-activated devices, the more trends, and patterns the system identifies based on the information it receives. Then, this data can be utilized to determine user preferences and tastes, which is a long-term selling point for making a home smarter. Google and Amazon are looking to integrate voice-enabled artificial intelligence capable of analyzing and responding to human emotion.

## 8.0 References/Bibliography

---

- <https://towardsdatascience.com/how-to-build-your-own-ai-personal-assistant-using-python-f57247b4494b?gi=337e33afb61b>
- <https://towardsdatascience.com/build-your-first-voice-assistant-%20%208%205a5a49f6cc1/%20h%20https://stackoverflow.com/>
- <https://www.geeksforgeeks.org/personal-voice-assistant-in-python/>
- <https://clearbridgemobile.com/7-key-predictions-for-the-future-of-voiceassistants/>

# Appendices

---

## Iron Man Jarvis AI Desktop Voice Assistant

```

import pyttsx3
import random
import re
import smtplib
import winsound
import wikipedia
import sys
import os
import webbrowser
import datetime
import speech_recognition as sr
from urllib.request import urlopen
from PyQt5 import QtWidgets, QtGui, QtCore
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.uic import loadUiType
from geopy import Nominatim
from pip._vendor import requests
from bs4 import BeautifulSoup as soup
from yahoo_fin import stock_info

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 wish():
    hour = int(datetime.datetime.now().hour)
    if hour >= 0 and hour < 12:
        speak("hello sir Good morning i am jarvis")
    elif hour >= 12 and hour < 18:
        speak("hello sir Good Afternoon i am jarvis")
    else:
        speak("hello sir Good evening i am jarvis")

def sendEmail(to, content):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login('naugraiyasuryansh@gmail.com', '6394682401')

```

```

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

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:
            speak("please tell me what to do")
            audio = R.listen(source)
        try:
            speak("wait Recognizing")
            text = R.recognize_google(audio, language='en-in')
        except Exception:
            return "none"
        text = text.lower()
        return text

    def JTT(self):
        R = sr.Recognizer()
        with sr.Microphone() as source:
            speak("listening")
            audio = R.listen(source)
        try:
            speak("wait Recognizing")
            text1 = R.recognize_google(audio, language='en-in')
        except Exception:
            speak("not able to recognize")
            return ""
        text1 = text1.lower()
        return text1

    def JARVIS(self):
        speak("starting")
        speak("initiating the system")
        speak("loading")
        wish()
        while True:

            self.query = self.STT()

            if ('bye' in self.query or 'quit' in self.query or 'bye
bye' in self.query or 'leave' in self.query or "nothing" in self.query or
"shutdown" in self.query):
                speak('Bye Sir. Have a nice day')
                winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
                sys.exit()

            elif ("hello" in self.query):
                hour = int(datetime.datetime.now().hour)
                if hour >= 0 and hour < 12:
                    speak("hello sir Good morning")
                elif hour >= 12 and hour < 18:
                    speak("hello sir Good Afternoon")
                else:

```



```

        speak("hello sir Good evening")

        elif ('music' in self.query or 'refresh' in self.query or
'anything' in self.query or 'song' in self.query):
            if('music' in self.query or 'song' in self.query):
                speak("wait i am playing music")
            else:
                speak("ok then i am playing music to refresh ur
mind")

                self.music_dir = "D:\\SONGS\\English Songs"
                self.musics = os.listdir(self.music_dir)
                num = random.randint(0, 11)

os.startfile(os.path.join(self.music_dir, self.musics[num]))

        elif 'wikipedia' in self.query:
            self.query = self.query.replace("wikipedia", "")
            if self.query:
                speak("Searching Wikipedia...")
                self.results = wikipedia.summary(self.query,
sentences=2)

                speak("According to Wikipedia")
                speak(self.results)
            else:
                speak("sorry sir not recognizing")
                pass

        elif 'time' in self.query:
            strTime = datetime.datetime.now().strftime("%H:%M:%S")
            speak("The time is: ")
            speak(strTime)

        elif 'launch vs code' in self.query:
            speak("please wait i am launching v s code")
            self.codepath = "C:\\Program Files\\Microsoft VS
Code\\Code.exe"
            os.startfile(self.codepath)

        elif ('launch java') in self.query:
            speak("please wait i am launching intelli j")
            self.codepath = "C:\\Program Files\\JetBrains\\IntelliJ
IDEA 2020.1.1\\bin\\idea64.exe"
            os.startfile(self.codepath)

        elif ('launch notepad') in self.query:
            speak("please wait i am launching notepad")
            self.codepath =
"C:\\Users\\snaug\\AppData\\Roaming\\Microsoft\\Windows\\Start
Menu\\Programs\\Accessories\\notepad"
            os.startfile(self.codepath)

        elif 'email' in self.query:
            try:
                speak("What should i say?")
                self.content = self.STT()
                to = "snaugraiya10@gmail.com"
                sendEmail(to, self.content)
                speak("Email has been sent!")
            except Exception as e:
                winsound.PlaySound("SystemExit", winsound.SND_ALIAS)

```

```

        speak("Sorry! i m not able to send this email")
        pass

    elif 'open' in self.query:
        speak("please wait")
        self.reg_ex = re.search('open (.+)', self.query)
        if self.reg_ex:
            self.domain = self.reg_ex.group(1)
            if (self.domain == "youtube"):
                speak("what to search")
                self.line = self.JTT()
                if self.line:
                    speak("Hold on, searching " + self.line)
                    self.url =
"https://www.youtube.com/results?search_query=" + self.line
                    webbrowser.open(self.url)
                else:
                    speak("so opening youtube")
                    self.url = "https://www.youtube.com"
                    webbrowser.open(self.url)
            elif (self.domain == "google"):
                speak("what to search")
                self.line = self.JTT()
                if self.line:
                    speak("Hold on, searching " + self.line)
                    self.url = 'https://www.google.com/search?q='
+ self.line

                    webbrowser.open(self.url)
                else:
                    speak("so opening google")
                    self.url = "https://www.google.com"
                    webbrowser.open(self.url)
            else:
                self.url = "https://www." + self.domain + ".com"
                webbrowser.open(self.url)
                speak('The website you have requested has been
opened for you Sir.')
        else:
            winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
            speak("sorry sir not able to recognize")
            pass

    elif 'joke' in self.query:
        speak("please wait searching for joke")
        res = requests.get('https://icanhazdadjoke.com/',
headers={"Accept": "application/json"})
        if res.status_code == requests.codes.ok:
            speak(str(res.json()[ 'joke' ]))
            winsound.PlaySound('hahaha.wav',
winsound.SND_FILENAME|winsound.SND_NOWAIT)

        else:
            winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
            speak('oops!I ran out of jokes')
            pass

    elif 'news' in self.query:
        speak("please wait sir searching for latest news")
        try:
            news_url = "https://news.google.com/news/rss"

```

```

Client = urlopen(news_url)
xml_page = Client.read()
Client.close()
soup_page = soup(xml_page, "xml")
news_list = soup_page.findAll("item")
speak("latest news is")
for news in news_list[:2]:
    print(news.title.text)
    print("-" * 60)
    speak(news.title.text.encode('utf-8'))
except Exception as e:
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
    speak("sorry sir not able to get the news")
    pass

elif ('weather' in self.query or 'temperature' in
self.query):
    api_key = "54b1b9aeefe8b01a10b622d47828ef3d"
    base_url =
"http://api.openweathermap.org/data/2.5/weather?"
    speak("which city sir")
    city_name = self.JTT()
    if city_name:
        complete_url = base_url + "appid=" + api_key + "&q="
+ city_name

        response = requests.get(complete_url)
        x = response.json()
        if x["cod"] != "404":
            y = x["main"]
            c_temperature = y["temp"] - 272.15
            current_temperature =
"{:.2f}".format(c_temperature)
            z = x["weather"]
            weather_description = z[0]["description"]
            speak(" Temperature (in celcius unit) = " +
                str(current_temperature) +
                "\n description = " +
                str(weather_description))
        else:
            winsound.PlaySound("SystemExit",
winsound.SND_ALIAS)

            speak(" City Not Found Sir ")
            pass
    else:
        pass

elif 'help me' in self.query:
    speak("""
        You can use these commands and I'll help you out:
        1.
        2. Open xyz : replace xyz with any website name
        3. email : Follow up questions such as recipient
name, content will be asked in order.
        4. weather/temperature : Tells you the current
condition and temperture
        5.
        6.
        7.
        8. news : top news of today
        9. time : Current system time
    """)

```

```

10.
11."
")

elif ('city' in self.query or 'state' in self.query or
'location' in self.query):
    geolocator = Nominatim(user_agent="geoapiExercises")
    speak("PLease tell city or state name")
    self.line = self.JTT()
    if self.line:
        location = geolocator.geocode(self.JTT())
        speak("Country location is:" + str(location))
    else:
        pass

elif "google" in self.query:
    reg_ex = re.search("google (.+)", self.query)
    if reg_ex:
        search = reg_ex.group(1)
        speak("Hold on, I will search " + search)
        url = 'https://www.google.com/search?q=' + search
        chrome_path = "C:\\Program
Files\\Google\\Chrome\\Application\\chrome.exe"
        webbrowser.register('chrome', None,
webbrowser.BackgroundBrowser(chrome_path))
        webbrowser.get('chrome').open_new_tab(url)
    else:
        speak("sorry sir not able to recognize")
        pass

elif "stock" in self.query:
    stock_info.get_live_price("AMZN")
    speak("stock price of amazon is")
    speak(str(stock_info.get_live_price("AMZN")))

elif ("about yourself" in self.query or "who are you" in
self.query):
    speak("hi, i m jarvis. i m the personal assistant of
suryansh sir. i m multitalented")

elif ("your age" in self.query or "old" in self.query):
    speak("i am immortal sir")

elif ('good' in self.query or 'keep it up' in self.query or
'well done' in self.query):
    speak("thank u sir")

elif "none" in self.query:
    pass

else:
    winsound.PlaySound("SystemExit", winsound.SND_ALIAS)
    speak("sorry sir not able to recognize")

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_8 = QLabel
self.exitB.setStyleSheet("background-
image:url(./lib/exit.png);\n"
"border:none;")
self.exitB.clicked.connect(self.close)
self.setWindowFlags(flags)
Dspeak = mainT()
self.label_8 = QMovie("./lib/initiating system.gif",
QByteArray(), self)
self.label_8.setCacheMode(QMovie.CacheAll)
self.label_6.setMovie(self.label_8)
self.label_8.start()
self.label_9 = QMovie("./lib/loading.gif", QByteArray(), self)
self.label_9.setCacheMode(QMovie.CacheAll)
self.label_7.setMovie(self.label_9)
self.label_9.start()
Dspeak.start()
self.label.setPixmap(QPixmap("./lib/tuse.png"))

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