VIT®
BHOPAL
www.vitbhopal.ac.in

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

Kothrikalan, Sehore

Madhya Pradesh – 466114

15th February, 2023

AI ASSISTANT

PROJECT REPORT

NAME OF THE CANDIDATES

ABHISHEK SINGH (21BAI10164)

ABHISHEK KUMAR (21BAI10021)

SATYAM KUMAR (21BAI10132)

SHANU SHAWANT (21BAI10083)

BACHELOR OF TECHNOLOGY

Computer Science

PROGRAM OF STUDY

Specialization in Artificial Intelligence and Machine Learning

VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE

MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled "Virtual AI Assistant" is the work of Abhishek Singh(21BAI10164), Satyam Kumar (21BAI10132), Abhishek Kumar (21BAI10021), Shanu Shawant (21BAI10083) who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR

Dr. S. Suthir

(Senior Assistant Professor)

School of Computer Science and Engineering

VIT BHOPAL UNIVERSITY

PROJECT GUIDE

Dr. Manoj Kumar

(Senior Assistant Professor)

School of Computer Science and Engineering

VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on _____

ACKNOWLEDGEMENT

First and foremost, we would like to thank the Lord Almighty for his presence and immense blessings throughout the project work.

We wish to express our heartfelt gratitude to Dr S. Suthir, Head of the Department, for giving us the golden opportunity to present out ideas through this means of project work.

We would like to thank my mentor Dr. Manoj kumar , for continually guiding and supporting us throughout the duration of the project work.

Last, but not least, we are deeply indebted by our parents who have been the greatest support while we worked day and night for the project to make it a success.

TABLE OF CONTENTS

PROJECT DESCRIPTION AND OUTLINE

An intelligent virtual assistant (IVA) or intelligent personal assistant (IPA) is a software agent that can perform tasks or services for an individual based on commands or questions. The term "chatbot" is sometimes used to refer to virtual assistants generally or specifically accessed by online chat. In some cases, online chat programs are exclusively for entertainment purposes. Some virtual assistants are able to interpret 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, and calendars with verbal commands.

One of the best-known virtual assistants is Apple's Siri, a consumer-facing product packaged as a personal assistant. Examples of other IVAs include Amazon's Alexa, Microsoft's Cortana, and Google's Google Assistant. Siri and competitors help customers easily execute commands with voice prompts, automating tasks such as setting alarms on a smartphone, verbally reading out e-mails with text-to-speech technology, playing and searching for music, and sending text messages. The ubiquity and popularity of IVAs in consumer smartphones led to the inclusion of Intelligent personal assistant technology by car manufacturers.

Types of AI Virtual Assistants

Chatbots have been a mainstay of the E-commerce sector since their inception, but modern implementations of chatbots are powered by artificial intelligence, which gives them the ability to think through customer queries rather than push the customer through a chain of static events

Voice assistants use automatic speech recognition and natural language processing to give vocal responses to queries, such as the well-known Siri and Google Assistant products.

AI avatars are 3D models designed to look like humans, used for entertainment applications, or to give a human touch to virtual customer support interactions. Cutting-edge technology from companies like Nvidia is capable of producing nearly true-to-life human avatars in real-time.

Domain-specific virtual assistants are highly specialized implementations of AI virtual assistants designed for very specific industries, optimized for high performance in travel, finance, engineering, cybersecurity, and other demanding sectors.

WORK RELATED INVESTIGATION

Some facts about AI ASSISTANT: -

Big Companies Will Most Likely Implement an AI Strategy

Artificial Intelligence stats from MIT Sloan show that 75 percent of top executives believe that AI will allow their organization to grow and achieve a competitive edge.

Most Consumers Think That AI Will Improve Their Lives

According to a survey by Strategy Analytics, 41 percent of the respondents in India, China, Western Europe, and the United States feel that emerging AI technologies will create a better life for them.

A Lot of People Are Unaware That They Use AI Platforms

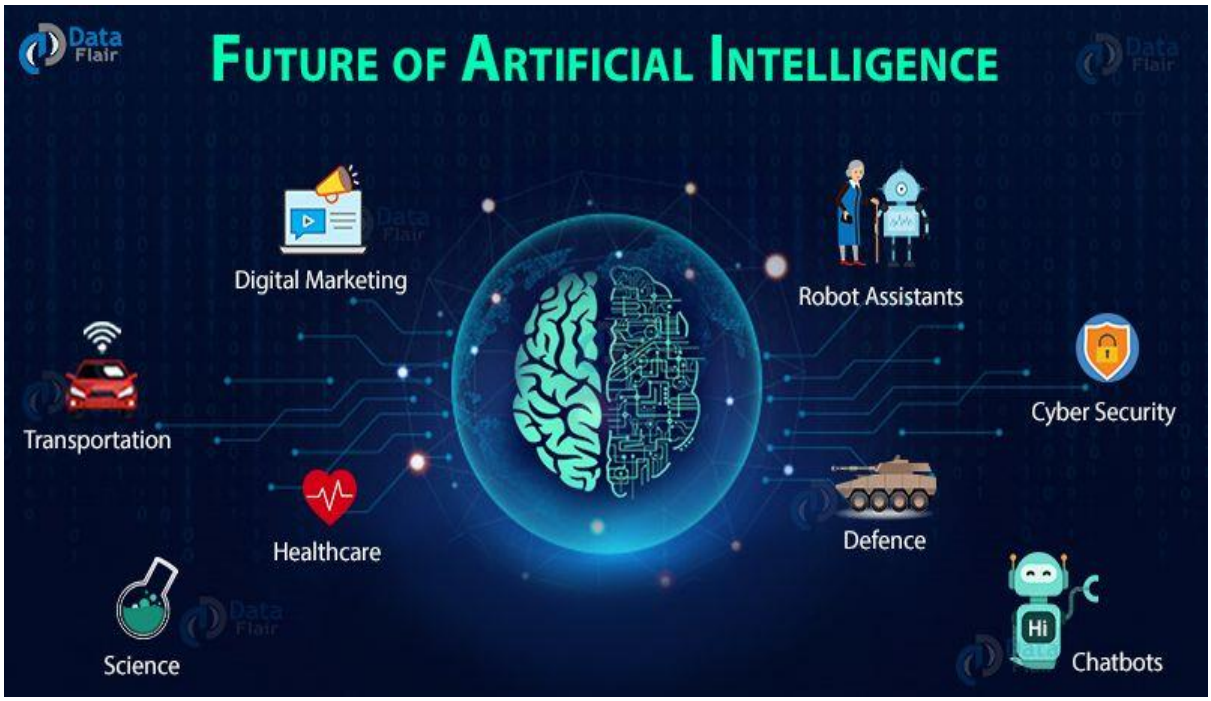
One of the strange Artificial Intelligence facts is that only 34% of consumers realize that they are directly experiencing AI, according to a study published by Pegasystems Inc. However, when surveyed about technologies they use, it was found that 84% actually use one or more AI-powered devices or services.

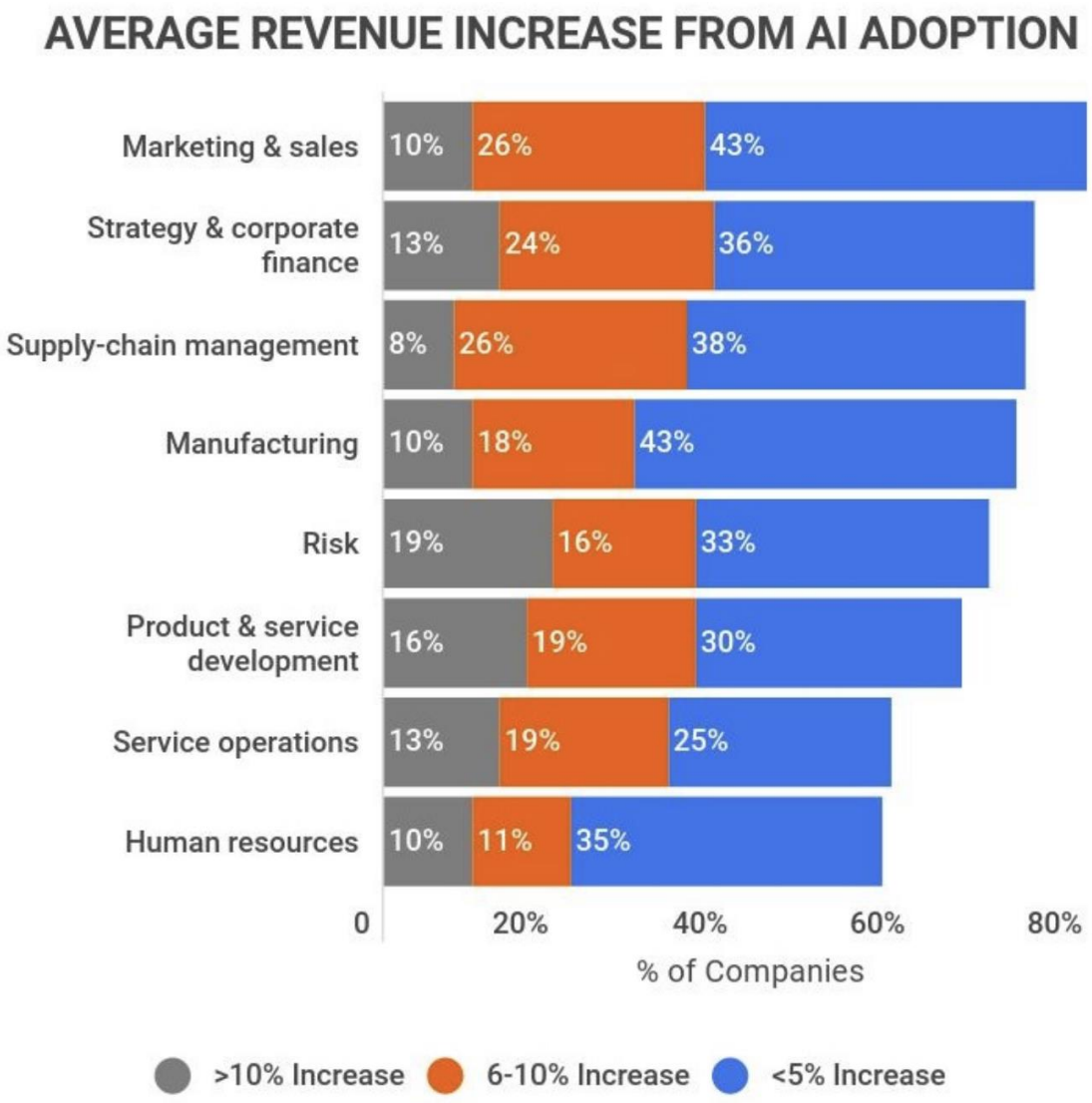
Almost All Smartphone Users Take Advantage of AI Voice Assistants

According to a study by Creative Strategies, 96 percent of Android consumers and 98 percent of iPhone consumers use Google and Apple's AI-based digital assistants - 'OK Google' and 'Siri'. The study further highlights that 51 percent of consumers use digital assistants in cars, 39 percent in homes, 6 percent in public places, and 1.3 percent at work.

Voice-Search Feature Is Gaining Widespread Popularity

AI-powered voice-search feature on smartphones, smart speakers, and other voice-enabled gadgets is becoming increasingly mainstream, thanks to the technological advancements in the field of speech-recognition. Current Artificial Intelligence stats point out that 41 percent of people who use smart devices utilize the voice-search feature as often as once a day.





TECHNOLOGIES AND TECH STACKS USED

We built a system using python programming language and its different modules and functions. We have also created a GUI interface for interacting to the user and properly taking and replying to the commands with the help of Qt Designer App.

Modules used :-

PYTTSX3	DATETIME
SMTPLIBAIO	PYAUTOGUI
WIKIPEDIA	PYJOKES
REQUESTS	PYWHATKIT
STRING COMPARISON	PYAUTOGUI

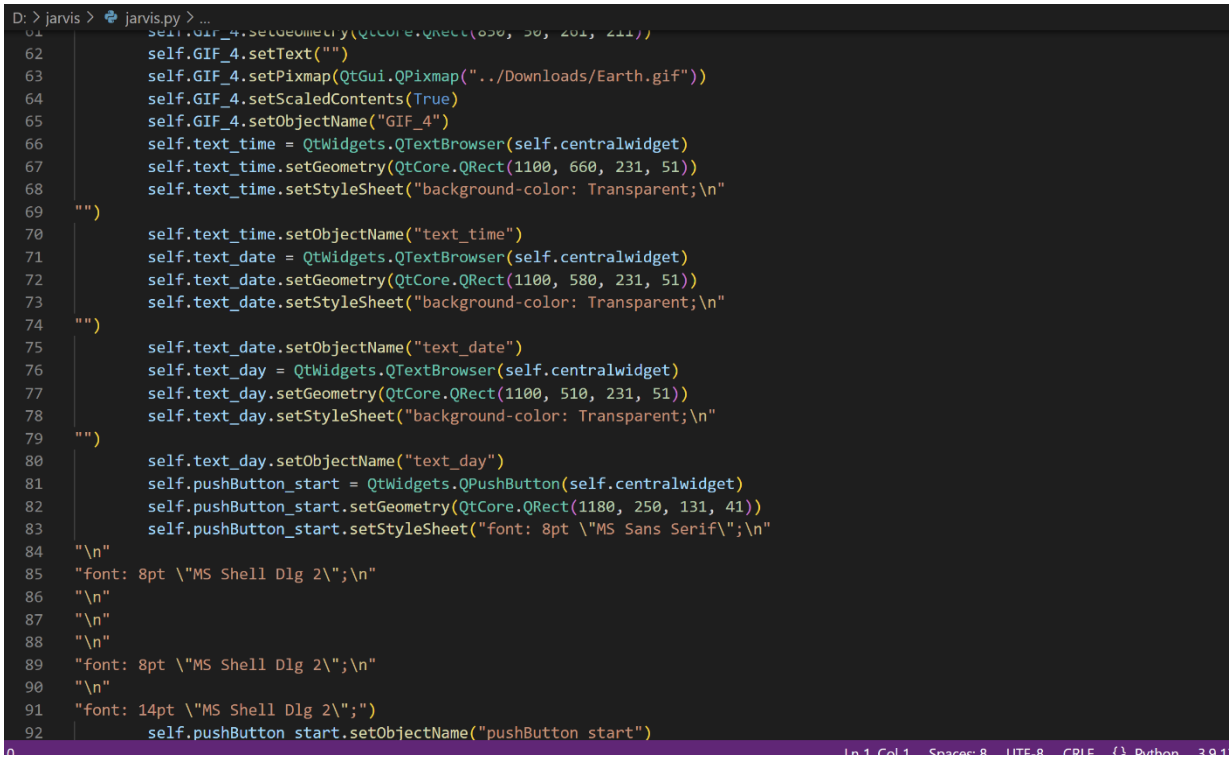
Qt designer code

D: > jarvis > jarvis.py > ...

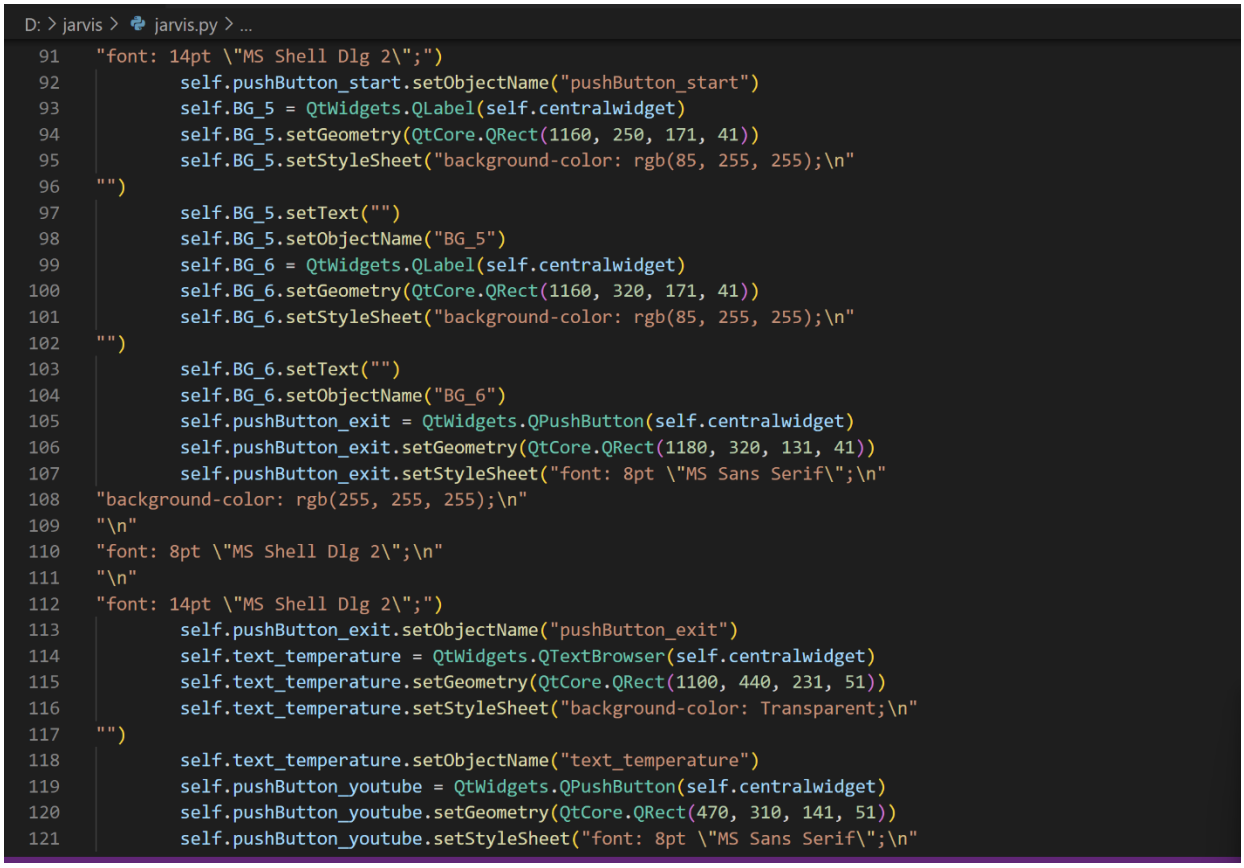
```
        self.BG_2.setObjectName("BG_2")
        self.BG_3 = QtWidgets.QLabel(self.centralwidget)
        self.BG_3.setGeometry(QtCore.QRect(420, 40, 381, 231))
        self.BG_3.setStyleSheet("background-color: rgb(85, 255, 255);")
        self.BG_3.setText("")
        self.BG_3.setObjectName("BG_3")
        self.GIF_1 = QtWidgets.QLabel(self.centralwidget)
        self.GIF_1.setGeometry(QtCore.QRect(40, 50, 311, 211))
        self.GIF_1.setText("")
        self.GIF_1.setPixmap(QtGui.QPixmap("../Downloads/Iron_Template_1.gif"))
        self.GIF_1.setScaledContents(True)
        self.GIF_1.setObjectName("GIF_1")
        self.GIF_2 = QtWidgets.QLabel(self.centralwidget)
        self.GIF_2.setGeometry(QtCore.QRect(430, 50, 361, 211))
        self.GIF_2.setText("")
        self.GIF_2.setPixmap(QtGui.QPixmap("../Downloads/7LP8.gif"))
        self.GIF_2.setScaledContents(True)
        self.GIF_2.setObjectName("GIF_2")
        self.GIF_3 = QtWidgets.QLabel(self.centralwidget)
        self.GIF_3.setGeometry(QtCore.QRect(30, 230, 391, 311))
        self.GIF_3.setText("")
        self.GIF_3.setPixmap(QtGui.QPixmap("../Downloads/__1.gif"))
        self.GIF_3.setScaledContents(True)
        self.GIF_3.setObjectName("GIF_3")
        self.BG_4 = QtWidgets.QLabel(self.centralwidget)
        self.BG_4.setGeometry(QtCore.QRect(840, 40, 281, 231))
        self.BG_4.setStyleSheet("background-color: rgb(85, 255, 255);")
        self.BG_4.setText("")
        self.BG_4.setObjectName("BG_4")
        self.GIF_4 = QtWidgets.QLabel(self.centralwidget)
        self.GIF_4.setGeometry(QtCore.QRect(850, 50, 261, 211))
```

D: > jarvis > jarvis.py > ...

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'jarvis.ui'
#
# Created by: PyQt5 UI code generator 5.15.4
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1344, 696)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.BG_1 = QtWidgets.QLabel(self.centralwidget)
        self.BG_1.setGeometry(QtCore.QRect(-750, -110, 2311, 991))
        self.BG_1.setText("")
        self.BG_1.setPixmap(QtGui.QPixmap("../Downloads/Black_Template.jpg"))
        self.BG_1.setScaledContents(True)
        self.BG_1.setObjectName("BG_1")
        self.BG_2 = QtWidgets.QLabel(self.centralwidget)
        self.BG_2.setGeometry(QtCore.QRect(30, 40, 331, 231))
        self.BG_2.setStyleSheet("background-color: rgb(73, 220, 220);\n"
"background-color: rgb(85, 255, 255);")
        self.BG_2.setText("")
        self.BG_2.setObjectName("BG_2")
```



```
D: > jarvis > jarvis.py > ...
        self.GIF_4.setGeometry(QtCore.QRect(850, 50, 261, 211))
        self.GIF_4.setText("")
        self.GIF_4.setPixmap(QtGui.QPixmap("../Downloads/Earth.gif"))
        self.GIF_4.setScaledContents(True)
        self.GIF_4.setObjectName("GIF_4")
        self.text_time = QtWidgets.QTextBrowser(self.centralwidget)
        self.text_time.setGeometry(QtCore.QRect(1100, 660, 231, 51))
        self.text_time.setStyleSheet("background-color: Transparent;\n"
"")
        self.text_time.setObjectName("text_time")
        self.text_date = QtWidgets.QTextBrowser(self.centralwidget)
        self.text_date.setGeometry(QtCore.QRect(1100, 580, 231, 51))
        self.text_date.setStyleSheet("background-color: Transparent;\n"
"")
        self.text_date.setObjectName("text_date")
        self.text_day = QtWidgets.QTextBrowser(self.centralwidget)
        self.text_day.setGeometry(QtCore.QRect(1100, 510, 231, 51))
        self.text_day.setStyleSheet("background-color: Transparent;\n"
"")
        self.text_day.setObjectName("text_day")
        self.pushButton_start = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_start.setGeometry(QtCore.QRect(1180, 250, 131, 41))
        self.pushButton_start.setStyleSheet("font: 8pt \"MS Sans Serif\";\n"
"\n"
"font: 8pt \"MS Shell Dlg 2\";\n"
"\n"
"\n"
"\n"
"font: 8pt \"MS Shell Dlg 2\";\n"
"\n"
"font: 14pt \"MS Shell Dlg 2\";")
        self.pushButton_start.setObjectName("pushButton_start")
```



```
D: > jarvis > jarvis.py > ...
"font: 14pt \"MS Shell Dlg 2\";")
        self.pushButton_start.setObjectName("pushButton_start")
        self.BG_5 = QtWidgets.QLabel(self.centralwidget)
        self.BG_5.setGeometry(QtCore.QRect(1160, 250, 171, 41))
        self.BG_5.setStyleSheet("background-color: rgb(85, 255, 255);\n"
"")
        self.BG_5.setText("")
        self.BG_5.setObjectName("BG_5")
        self.BG_6 = QtWidgets.QLabel(self.centralwidget)
        self.BG_6.setGeometry(QtCore.QRect(1160, 320, 171, 41))
        self.BG_6.setStyleSheet("background-color: rgb(85, 255, 255);\n"
"")
        self.BG_6.setText("")
        self.BG_6.setObjectName("BG_6")
        self.pushButton_exit = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_exit.setGeometry(QtCore.QRect(1180, 320, 131, 41))
        self.pushButton_exit.setStyleSheet("font: 8pt \"MS Sans Serif\";\n"
"background-color: rgb(255, 255, 255);\n"
"\n"
"font: 8pt \"MS Shell Dlg 2\";\n"
"\n"
"font: 14pt \"MS Shell Dlg 2\";")
        self.pushButton_exit.setObjectName("pushButton_exit")
        self.text_temperature = QtWidgets.QTextBrowser(self.centralwidget)
        self.text_temperature.setGeometry(QtCore.QRect(1100, 440, 231, 51))
        self.text_temperature.setStyleSheet("background-color: Transparent;\n"
"")
        self.text_temperature.setObjectName("text_temperature")
        self.pushButton_youtube = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton_youtube.setGeometry(QtCore.QRect(470, 310, 141, 51))
        self.pushButton_youtube.setStyleSheet("font: 8pt \"MS Sans Serif\";\n"
```

D: > jarvis > jarvis.py > ...

```
122     "background-color: rgb(85, 255, 255);\n"
123     "font: 16pt \"MS Shell Dlg 2\";")
124             self.pushButton_youtube.setObjectName("pushButton_youtube")
125             self.pushButton_chrome = QtWidgets.QPushButton(self.centralwidget)
126             self.pushButton_chrome.setGeometry(QtCore.QRect(650, 310, 141, 51))
127             self.pushButton_chrome.setStyleSheet("font: 8pt \"MS Sans Serif\";\n"
128     "font: 14pt \"MS Shell Dlg 2\";\n"
129     "\n"
130     "background-color: rgb(85, 255, 255);")
131             self.pushButton_chrome.setObjectName("pushButton_chrome")
132             self.pushButton_whatsapp = QtWidgets.QPushButton(self.centralwidget)
133             self.pushButton_whatsapp.setGeometry(QtCore.QRect(830, 310, 141, 51))
134             self.pushButton_whatsapp.setStyleSheet("font: 8pt \"MS Sans Serif\";\n"
135     "font: 14pt \"MS Shell Dlg 2\";\n"
136     "background-color: rgb(85, 255, 255);")
137             self.pushButton_whatsapp.setObjectName("pushButton_whatsapp")
138             self.BG_1.raise_()
139             self.GIF_3.raise_()
140             self.BG_5.raise_()
141             self.BG_2.raise_()
142             self.BG_3.raise_()
143             self.GIF_1.raise_()
144             self.GIF_2.raise_()
145             self.BG_4.raise_()
146             self.GIF_4.raise_()
147             self.text_time.raise_()
148             self.text_date.raise_()
149             self.text_day.raise_()
150             self.BG_6.raise_()
151             self.pushButton_exit.raise_()
152             self.text_temperature.raise_()
```

D: > jarvis > jarvis.py > ...

```
153         self.pushButton_youtube.raise_()
154         self.pushButton_chrome.raise_()
155         self.pushButton_whatsapp.raise_()
156         self.pushButton_start.raise_()
157         MainWindow.setCentralWidget(self.centralwidget)
158         self.statusBar = QtWidgets.QStatusBar(MainWindow)
159         self.statusBar.setObjectName("statusBar")
160         MainWindow.setStatusBar(self.statusBar)
161 
162         self.retranslateUi(MainWindow)
163         QtCore.QMetaObject.connectSlotsByName(MainWindow)
164 
165     def retranslateUi(self, MainWindow):
166         _translate = QtCore.QCoreApplication.translate
167         MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
168         self.pushButton_start.setText(_translate("MainWindow", "Start"))
169         self.pushButton_exit.setText(_translate("MainWindow", "Exit"))
170         self.pushButton_youtube.setText(_translate("MainWindow", "Youtube"))
171         self.pushButton_chrome.setText(_translate("MainWindow", "Chrome"))
172         self.pushButton_whatsapp.setText(_translate("MainWindow", "Whatsapp"))
173 
174 
175 if __name__ == "__main__":
176     import sys
177     app = QtWidgets.QApplication(sys.argv)
178     MainWindow = QtWidgets.QMainWindow()
179     ui = Ui_MainWindow()
180     ui.setupUi(MainWindow)
181     MainWindow.show()
182     sys.exit(app.exec_())
183 
```

GUI code

D: > jarvis > jarvisgui.py > ...

```python
from threading import main_thread
from xml.dom.pulldom import START_ELEMENT
from jarvis import Ui_MainWindow
from PyQt5 import QtCore , QtGui , QtWidgets
from PyQt5.QtCore import *
from PyQt5.QtGui import QMovie
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import Qt, QTimer , QTime ,QDate
from PyQt5.uic import loadUiType
import main
import os
import webbrowser as web
import sys


class Mainthread(QThread):

    start_Exe = main_thread()

class Gui_start(QMainWindow):
    def __init__(self):

        super()._init_()


        self.Gui = Ui_MainWindow()
        self.setupUi(self)

        self.Gui.pushButton_start.clicked.connect(self.startTask)
        self.Gui.pushButton_exit.clicked.connect(self.close)
```

D: > jarvis > jarvisgui.py > ...

```
        self.Gui.pushButton_chrome.clicked.connect(self.chrome_app)
        self.Gui.pushButton_whatsapp.clicked.connect(self.whatsapp_app)
        self.Gui.pushButton_youtube.clicked.connect(self.youtube_app)

    def chrome_app(self):
        os.startfile("C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe")

    def youtube_app(self):
        web.open("https://www.youtube.com/")

    def whatsapp_app(self):
        web.open("https://web.whatsapp.com/")

    def startTask(self):
        self.Gui.label1 =QtGui.QMovie("../Downloads/Iron_Template_1.gif")
        self.Gui.GIF_1.setMovie(self.Gui.label1)
        self.Gui.label1.start()

        self.Gui.label2 =QtGui.QMovie("../Downloads/7LP8.gif")
        self.Gui.GIF_2.setMovie(self.Gui.label2)
        self.Gui.label2.start()

        self.Gui.label3 =QtGui.QMovie("../Downloads/__1.gif")
        self.Gui.GIF_3.setMovie(self.Gui.label3)
        self.Gui.label3.start()

        self.Gui.label4 =QtGui.QMovie("../Downloads/Earth.gif")
        self.Gui.GIF_4.setMovie(self.Gui.label4)
        self.Gui.label4.start()

        timer =QTimer(self)
```

Main code

D: > jarvis > main.py > ...

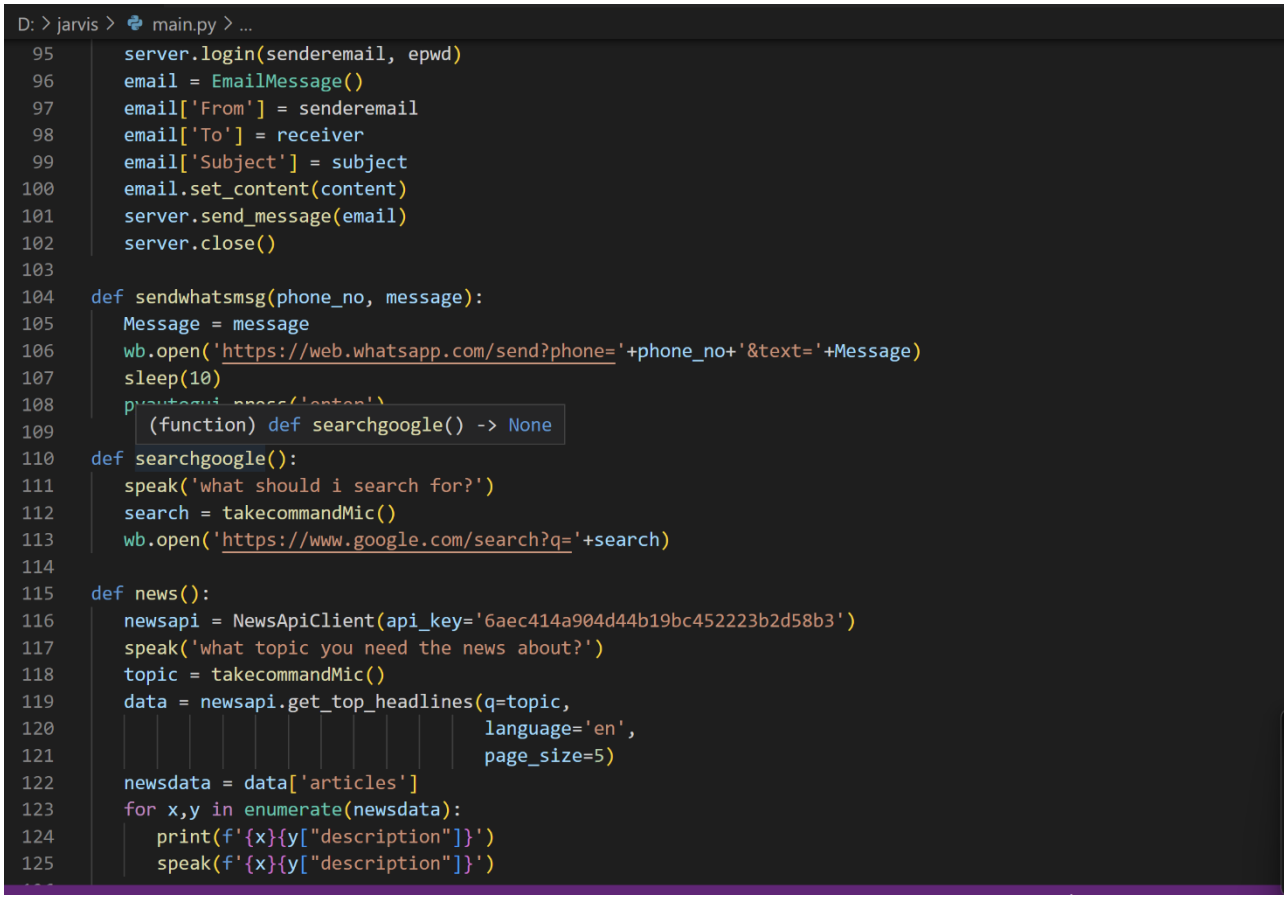
```python
import pyttsx3  #pip install pyttsx3 == text data into speech using python
import datetime
import speech_recognition as sr  #pip install SpeechRecongnition == speech d=from mic to text
import smtplib
from secrets import senderemail , epwd , to
from email.message import EmailMessage
import pyautogui
import webbrowser as wb
from time import sleep
import wikipedia
import pywhatkit
import requests
from newsapi import NewsApiClient
import clipboard
import os
import pyjokes
import time as tt
import string
import random
import psutil

engine = pyttsx3.init()

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

def getvoices(voice):
    voices = engine.getProperty('voices')
    #print(voices[1].id)
    if voice == 1:
```

D: > jarvis > main.py > ...

```
    if voice == 1:
        engine.setProperty('voice',voices[0].id)
        speak("hello this is jarvis ")

    if voice == 2:
        engine.setProperty('voice',voices[1].id)
        speak("hello this is jarvis ")

def time():
    Time = datetime.datetime.now().strftime("%I:%M:%S") # hour=I minute=M seconds=S
    speak("the current time is:")
    speak(Time)

def date():
    year = int(datetime.datetime.now().year)
    month = int(datetime.datetime.now().month)
    date = int(datetime.datetime.now().day)
    speak("the current date is:")
    speak(date)
    speak(month)
    speak(year)

def greeting():
    hour = datetime.datetime.now().hour
    if hour >= 6 and hour <12:
        speak("Good Morning Sir!")
    elif hour >= 12 and hour <18:
        speak("Good Afternoon Sir!")
    elif hour >= 18 and hour <24:
        speak("Good Evening Sir!")
    else:
        speak("Good Night Sir!")
```

D: > jarvis > main.py > ...

```
def wishme():
    speak("Welcome back sir!")
    time()
    date()
    greeting()
    speak("jarvis at your service , please tell me how can i help u")


def takecommandCMD():
    query = input("please tell me how can i help you ?\n")
    return query

def takecommandMic():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)
    try:
        print("recognizning...")
        query = r.recognize_google(audio , language="en-IN")
        print(query)
    except Exception as e:
        print(e)
        speak("Say that again Please...")
        return"None"
    return query

def sendEmail(receiver, subject, content):
    server = smtplib.SMTP('smtp.gmail.com',587)
    server.starttls()
```


D: > jarvis > main.py > ...
server.login(senderemail, epwd)
email = EmailMessage()
email['From'] = senderemail
email['To'] = receiver
email['Subject'] = subject
email.set_content(content)
server.send_message(email)
server.close()
def sendwhatsmsg(phone_no, message):
Message = message
wb.open('https://web.whatsapp.com/send?phone='+phone_no+'&text='+Message)
sleep(10)
(function) def searchgoogle() -> None
def searchgoogle():
speak('what should i search for?')
search = takecommandMic()
wb.open('https://www.google.com/search?q='+search)
def news():
newsapi = NewsApiClient(api_key='6aec414a904d44b19bc452223b2d58b3')
speak('what topic you need the news about?')
topic = takecommandMic()
data = newsapi.get_top_headlines(q=topic,
language='en',
page_size=5)
newsdata = data['articles']
for x,y in enumerate(newsdata):
print(f'{x}{y["description"]}')
speak(f'{x}{y["description"]}')

D: > jarvis > main.py > ...

```
    speak("that's it for now i'll update you in some time ")

def text2speech():
    text = clipboard.paste()
    print(text)
    speak(text)

def covid():
    r = requests.get('https://coronavirus-19-api.herokuapp.com/all')
    data = r.json()
    covid_data = f'confirmed cases :{data["cases"]} \n Deaths :{data["deaths"]} \n Recovered :{data["recovered
    print(covid_data)
    speak(covid_data)

def screenshot():
    name_img = tt.time()
    name_img = 'D:\\jarvis 2.0\\screenshot\\{name_img}.png'
    img = pyautogui.screenshot(name_img)
    img.show()

def passwordgen():
    s1 = string.ascii_uppercase
    s2 = string.ascii_lowercase
    s3 = string.digits
    s4 = string.punctuation

    passlen = 8
    s = []
    s.extend(list(s1))
    s.extend(list(s2))
    s.extend(list(s3))
```

D: > jarvis > main.py > ...

```
    s.extend(list(s4))

    random.shuffle(s)
    newpass = ("".join(s[0:passlen]))
    print(newpass)
    speak(newpass)

def flip():
    speak('okay sir, flipping a coin')
    coin = ['heads','tails']
    toss = []
    toss.extend(coin)
    random.shuffle(toss)
    toss = ("".join(toss[0]))
    speak('i flipped the coin and you got'+toss)

def roll():
    speak("okay sir , rolling a die for you")
    die = ['1','2','3','4','5','6']
    roll = []
    roll.extend(die)
    random.shuffle(roll)
    roll = ("".join(roll[0]))
    speak("i rolled the die and you get "+roll)

def cpu():
    usage = str(psutil.cpu_percent())
    speak('CPU is at'+ usage)
    battery = psutil.sensors_battery()
    speak("Battery is at")
    speak(battery.percent)
```

D: > jarvis > main.py > ...

```
189
190 if _name_ == "_main_":
191
192     getvoices(0)
193     wishme()
194     while True:
195         query = takecommandMic().lower()
196         if 'time' in query:
197             time()
198
199         elif 'date' in query:
200             date()
201
202         elif 'email' in query:
203             email_list = {
204                 'test email':'sk3345164@gmail.com'
205             }
206             try:
207                 speak("To whom you want to send the mail?")
208                 name = takecommandMic()
209                 receiver = email_list[name]
210                 speak("what is the subject of the mail?")
211                 subject = takecommandMic()
212                 speak('what shoud i say?')
213                 content = takecommandMic()
214                 sendEmail(receiver,subject,content)
215                 speak("Email has been send")
216
217             except Exception as e:
218                 print(e)
219                 speak("unable to send the email")
```

D: > jarvis > main.py > ...

```
221         elif 'message' in query:
222             user_name = {
223                 'Jarvis':'9709938257'
224             }
225             try:
226                 speak("To whom you want to send the whats App message?")
227                 name = takecommandMic()
228                 phone_no = user_name[name]
229                 speak("what is the subject of the message?")
230                 message = takecommandMic()
231                 sendwhatsmsg(phone_no, message)
232                 speak("Message has been send")
233             except Exception as e:
234                 print(e)
235                 speak("unable to send the message")
236
237         elif 'wikipedia' in query:
238             speak("Searching on wikipedia...")
239             query = query.replace("wikipedia","")
240             result = wikipedia.summary(query, sentences=2)
241             print(result)
242             speak(result)
243
244         elif 'search' in query:
245             searchgoogle()
246
247         elif 'youtube' in query:
248             speak("what should i search on youtube?")
249             topic = takecommandMic()
250             pywhatkit.playonyt(topic)
251
```

D: > jarvis > main.py > ...

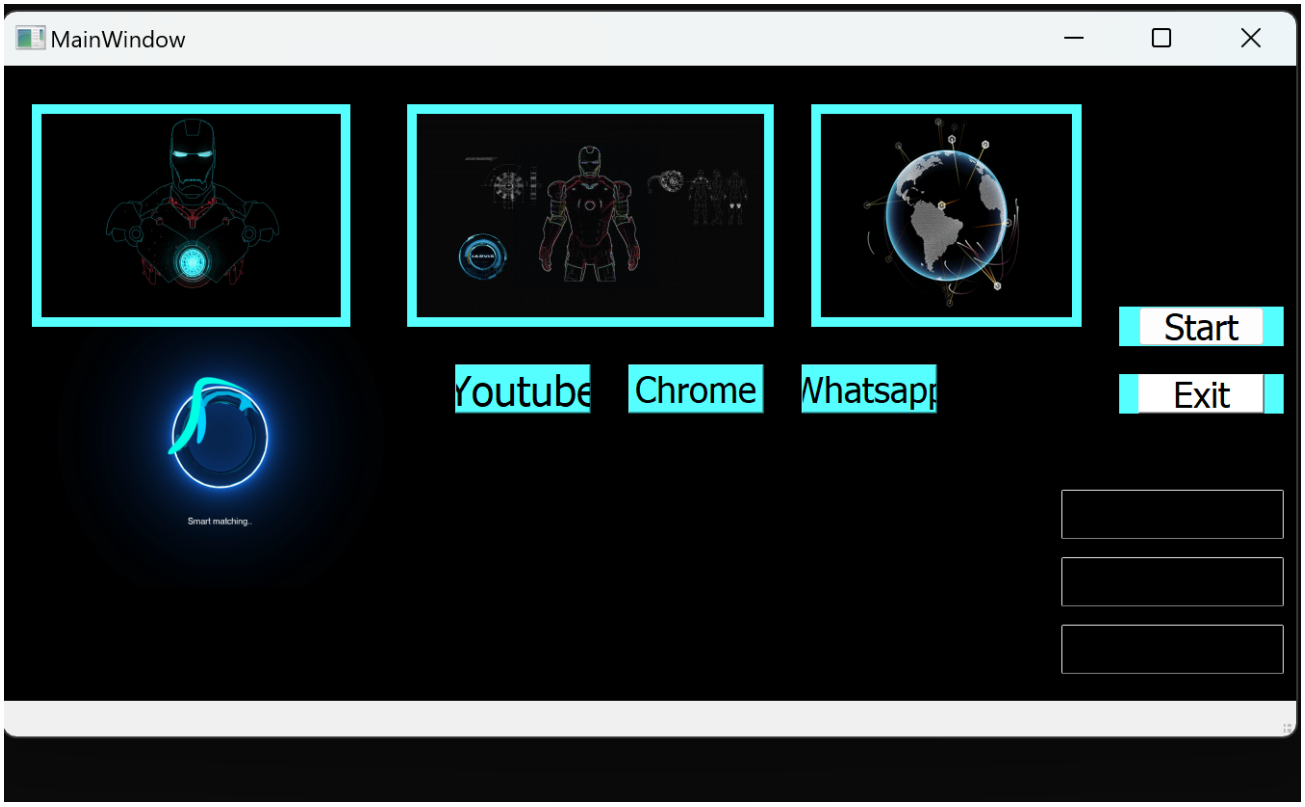
```
252     elif 'weather' in query:
253         city = 'new york'
254         url = f'http://api.openweathermap.org/data/2.5/weather?q={city}&units=imperial&appid=94a61c105d390b8
255         res  = requests.get(url)
256         data = res.json()
257 
258         weather = data['weather'] [0] ['main']
259         temp = data['main']['temp']
260         desp = data['weather'] [0]['description']
261         temp = round((temp-32 * 5/9))
262         print(weather)
263         print(temp)
264         print(desp)
265         speak(f'weather in {city} city is like')
266         speak('Temperature : {} degree'.format(temp))
267         speak('weather is {}'.format(desp))
268 
269     elif 'news' in query:
270         news()
271 
272     elif 'read' in query:
273         text2speech()
274 
275     elif 'covid' in query:
276         covid()
277 
278     elif 'open code' in query:
279         codepath = 'C:\\Users\\abhishek kumar\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe'
280         os.startfile(codepath)
281 
```

D: > jarvis > main.py > ...

```
281 
282     elif 'open' in query:
283         os.system('explorer c://{}'.format(query.replace('open','')))
284 
285     elif 'joke' in query:
286         speak(pyjokes.get_joke())
287 
288     elif 'screenshot' in query:
289         screenshot()
290         speak("done!")
291 
292     elif 'remember that' in query:
293         speak("what should i remember?")
294         data = takecommandMic()
295         speak("you said me to remember that"+data)
296         remember = open('data.txt','w')
297         remember.write(data)
298         remember.close()
299 
300     elif 'do you know anything' in query:
301         remember = open('data.txt','r')
302         speak('you said me to remember that'+remember.read())
303 
304     elif 'password' in query:
305         passwordgen()
306 
307     elif 'flip' in query:
308         flip()
309 
310     elif 'roll' in query:
311         roll()
```

```
D: > jarvis > main.py > ...
300         elif 'do you know anything' in query:
301             remember = open('data.txt','r')
302             speak('you said me to remember that'+remember.read())
303 
304         elif 'password' in query:
305             passwordgen()
306 
307         elif 'flip' in query:
308             flip()
309 
310         elif 'roll' in query:
311             roll()
312 
313         elif 'cpu' in query:
314             cpu()
315 
316         elif 'offline' in query:
317             quit()
```

GUI interface



REFERENCES

1. https://www.youtube.com/watch?v=Lp9Ftuq2sVI have taken some help from this video for coding parts.

2. https://www.youtube.com/watch?v=xfh0fCnfrpE have taken help from this video for Gui Formation.