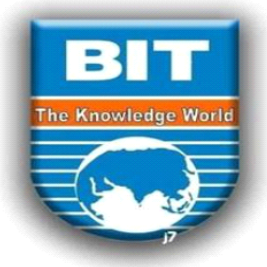
PROJECT REPORT

2020 - 2021



" VIRTUAL INTELLIGENT PERSONAL ASSSISTANT"

Submitted For

Diploma in Engineering
(Computer Engineering)

Maharashtra State Board Of Technical Education, Mumbai (M.S.)

Submitted By

MR. OMPRAKASH S. KHANDALE

MR. LOVELY K. SHARMA

Guided By

PROF. SANDHYA M. PARDHI
(Computer Engg. Department)

DEPARTMENT OF COMPUTER ENGINEERING
BTC - SCHOOL OF DIPLOMA IN ENGINEERING,
BALLARPUR (CHANDRAPUR)
(MS)

CERTIFICATE

This is to certify that the Project Entitled

" Virtual Intelligent Personal Assistant "

Has been successfully completed by

MR. OMPRAKASH S KHANDALE **MR. LOVELY K SHARMA**

In the partial fulfillment for the

Diploma in Engineering
(Computer Engineering)

Awarded By:

-

Maharashtra State Board Of Technical Education, Mumbai (M.S.)

During the academic year 2020 - 2021 under my guidance.

Project Guide

Prof. Sandhaya M. Pardhi
BE(CSE) , M.Tech(CSE)
(Comp. Engineering Dept. BTC,Ballarpur)

Prof. Arti A. Vaidya
BE(CSE) , M.Tech(CSE)
HEAD OF COMP.ENGG. DEPT.
BTC- School of Diploma in Engg. .

Prof. S.S.GOJE
BE(ETC), M.Tech(E.S)
PRINCIPAL
BTC- School of Diploma in Engg.

CERTIFICATE

This is to certify that the Project Entitled

" Virtual Intelligent Personal Assistant "

Has been successfully completed by

MR. OMPRAKASH S KHANDALE **MR. LOVELY K SHARMA**

In the partial fulfillment for the

Diploma in Engineering

(Computer Engineering)

Awarded By:

Maharashtra State Board Of Technical Education, Mumbai (M.S.)

During the academic year 2020 - 2021 under my guidance

Project Guide

Prof. Sandhya M.Pardhi
BE(CSE) , M.Tech(CSE)
(Comp. Engineering Dept. BTC,Ballarpur)

Prof. Arti .A.Vaidya
BE(CSE) , M.Tech(CSE)
HEAD OF COMP.ENGG. DEPT.
BTC- School of Diploma in Engg. .

Prof. S.S.GOJE
BE(ETC), M.Tech(E.S)
PRINCIPAL
BTC- School of Diploma in Engg.

EXAMINER'S CERTIFICATE

This is to certify that the following students

MR. OMPRAKASH S. KHANDALE **MR. LOVELY S. SHARMA**

Of
Diploma in Engineering
(Computer Engineering)

Were examined in the Project work entitled

"VIRTUAL INTELLEGENT PERSONAL ASSISTANT"

On / /2020 at BTC – School of Diploma in Engineering, Ballarpur.

Signature
Prof.
(Internal Examiner)
Date: / /2020

Signature
Prof.
(External Examiner)
Date: / /2020

DEPARTMENT OF COMPUTER ENGINEERING
BTC – SCHOOL OF DIPLOMA IN ENGINEERING, BALLARPUR
(CHANDRAPUR)

ACKNOWLEDHEMENT

We avail this opportunity to express our deep sense of gratitude and whole hearted thanks to our guide **Prof. Sandhya M.Pardhi** for their invaluable guidance and encouragement to embark this project.

We are also thankful to our Head of the Department, Computer Engineering, **Prof. A. A. Vaidya**, whose esteemed suggestions and encouragement from time to time have always been unparalleled stimuli for us to travel eventually towards completion of this project.

We are also thankful to our Honorable Principal **Prof. S. S. Goje** who inspired us a lot to achieve the highest goal.

Last but not the least we would like to thank all our friends who helped us directly or indirectly for success of our project.

Thank you

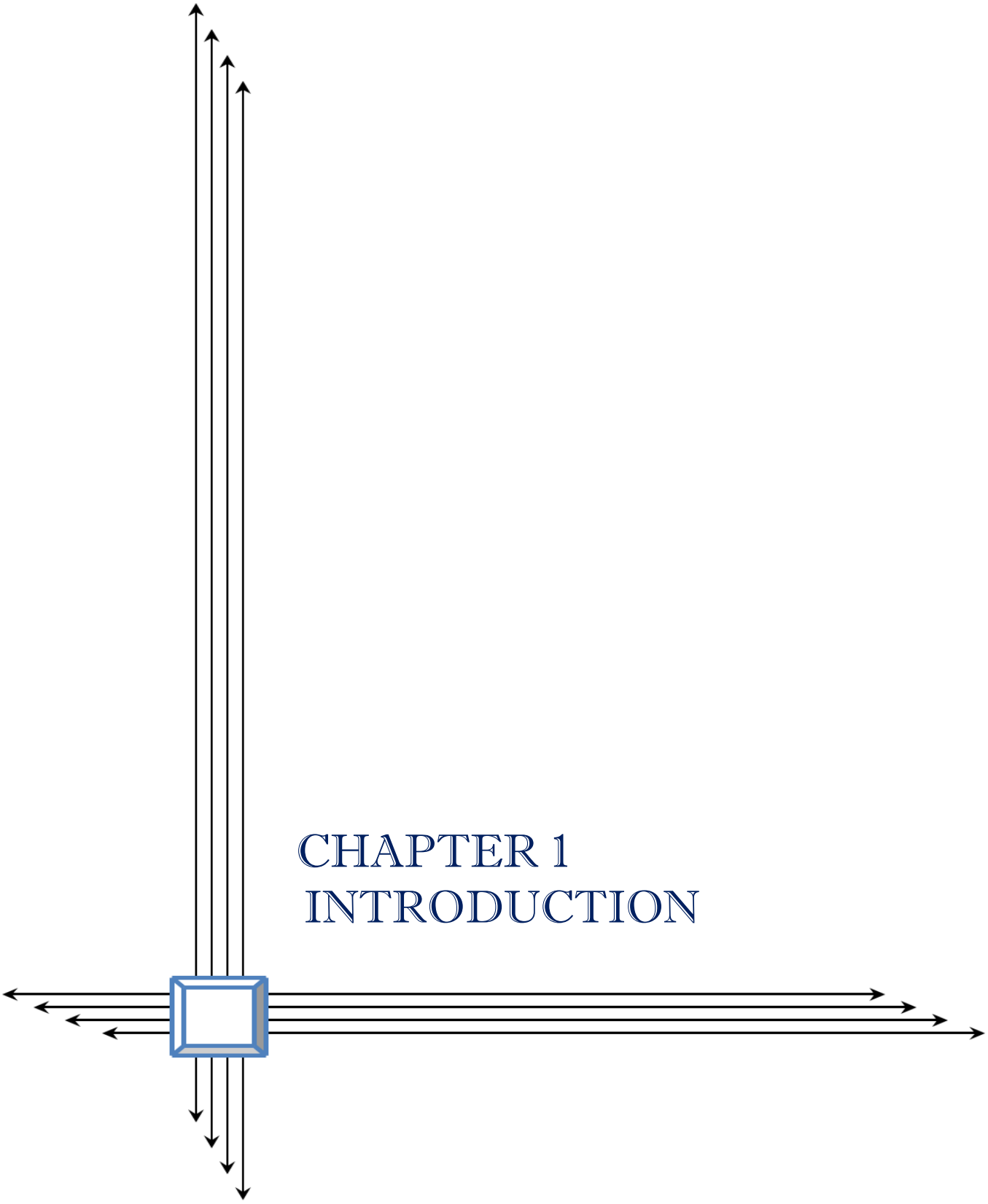
ABSTRACT

This project includes an implementation of an intelligent voice assistant for desktop. Where functionality on current existing applications on other platforms is compared. Until this day, there has not been any good alternative for Windows, so this project aims to implement a voice assistant for the Windows platform while describing the difficulties and challenges that lies in this task.

Voice assistants are software agents that can interpret human speech and respond via synthesized voices. Apple's Siri, Amazon's Alexa, Microsoft's Cortana, and Google's Assistant are the most popular voice assistants and are embedded in smartphones or dedicated home speakers. Users can ask their assistants questions, control home automation devices and media playback via voice, and manage other basic tasks such as to-do lists, checking weather, Google searching engine, Wikipedia searching engine, camera, and many more functionalities with verbal commands. This column will explore the basic workings and common features of today's voice assistants. It will also discuss some of the privacy and security issues inherent to voice assistants and some potential future uses for these devices.

The attempt has been made to develop an "Intelligent Personal Voice Assistant using Python" which helps people to control device with their voice (speech), extract information and perform tasks on their desktop.

INDEX



CHAPTER 1
INTRODUCTION

INTRODUCTION

In today's era almost all tasks are digitalized. We have Smartphone in hands and it is nothing less than having world at your fingertips. These days we are not even using fingers. We just speak about the task and it is done. There exist systems where we can say Text Dad, "I'll be late today." And the text is sent. That is the task of a Virtual Voice Assistant.

This system is designed to be used efficiently on desktops. Personal assistant software improves user productivity by managing routine tasks of the user and by providing information from online sources to the user. People always wanted to talk to computers almost from the moment the first computer was invented. Just a few decades ago, the idea of holding meaningful conversation with a computer seemed futuristic, but the technology to make voice interfaces useful and widely available is already here. Several consumer-level products developed in the last few years have brought inexpensive voice assistants into everyday use, and more features and platforms are being added all the time.

Users can do everything from asking simple informational questions to playing music and dialing their phone or turning lights on and off via voice control. This column will explore the basic workings and common features of today's voice assistants. It will also discuss some potential future uses for these devices.

Voice assistants comes in small packages and can perform a variety of task and actions after hearing a wake up word or command. They can play music, turn on the lights, they can answer any question and also place an order for you.

This project was started on the premise that there is sufficient amount of openly available data and information on the web that can be utilized to build a virtual assistant that has access to making intelligent decisions for routine user activities.

1.1 Objective of Project

Main objective of building personal assistant software (a virtual assistant) is using semantic data sources available on the web, user generated content and providing knowledge from knowledge databases. The main purpose of an intelligent virtual assistant is to answer questions that users may have.

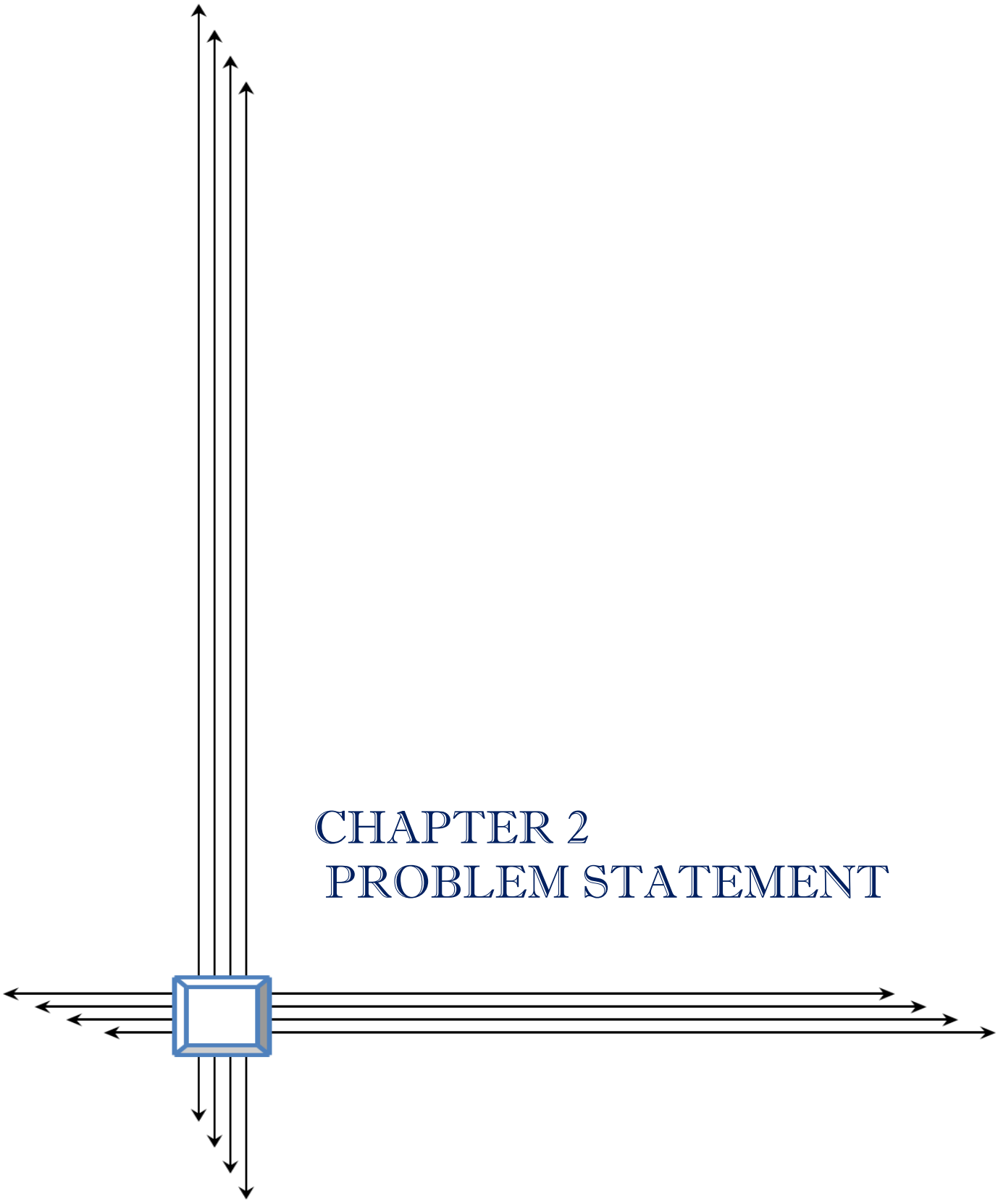
This project is focusing on the Desktop App development over the voice control (recognition, generate and analyze corresponding commands, intelligent responses automatically), Google products and relevant APIs (Google weather, Google search and etc), Wikipedia API and desktop device references ranging from Speech-To-Text, Text-To-Speech technology,

Virtual assistants can tremendously save you time. We spend hours in online research and then making the report in our terms of understanding. David can do that for you. Provide a topic for research and continue with your tasks while David does the research. Another difficult task is to remember test dates, birthdates or anniversaries. It comes with a surprise when you enter the class and realize it is class test today. Just tell David in advance about your tests and he reminds you well in advance so you can prepare for the test.

1.2 Limitations of Project

- They depend on an Internet connection.
- Voice assistants use single commands. For now, these consist mostly of fixed phrases. Effectively, they push one button or set one dial.
- As more flexible natural language understanding technology is becoming available, interpretations of speech commands may be become ambiguous. With commands resulting into actions, misunderstandings can be risky.
- Voice assistants cannot integrate context data, such as what I'm currently working on.

These shortcomings limit voice assistants from elevating their status from transactionary to really helpful. They need a semantic level of interaction to support more complex activities.



CHAPTER 2
PROBLEM STATEMENT

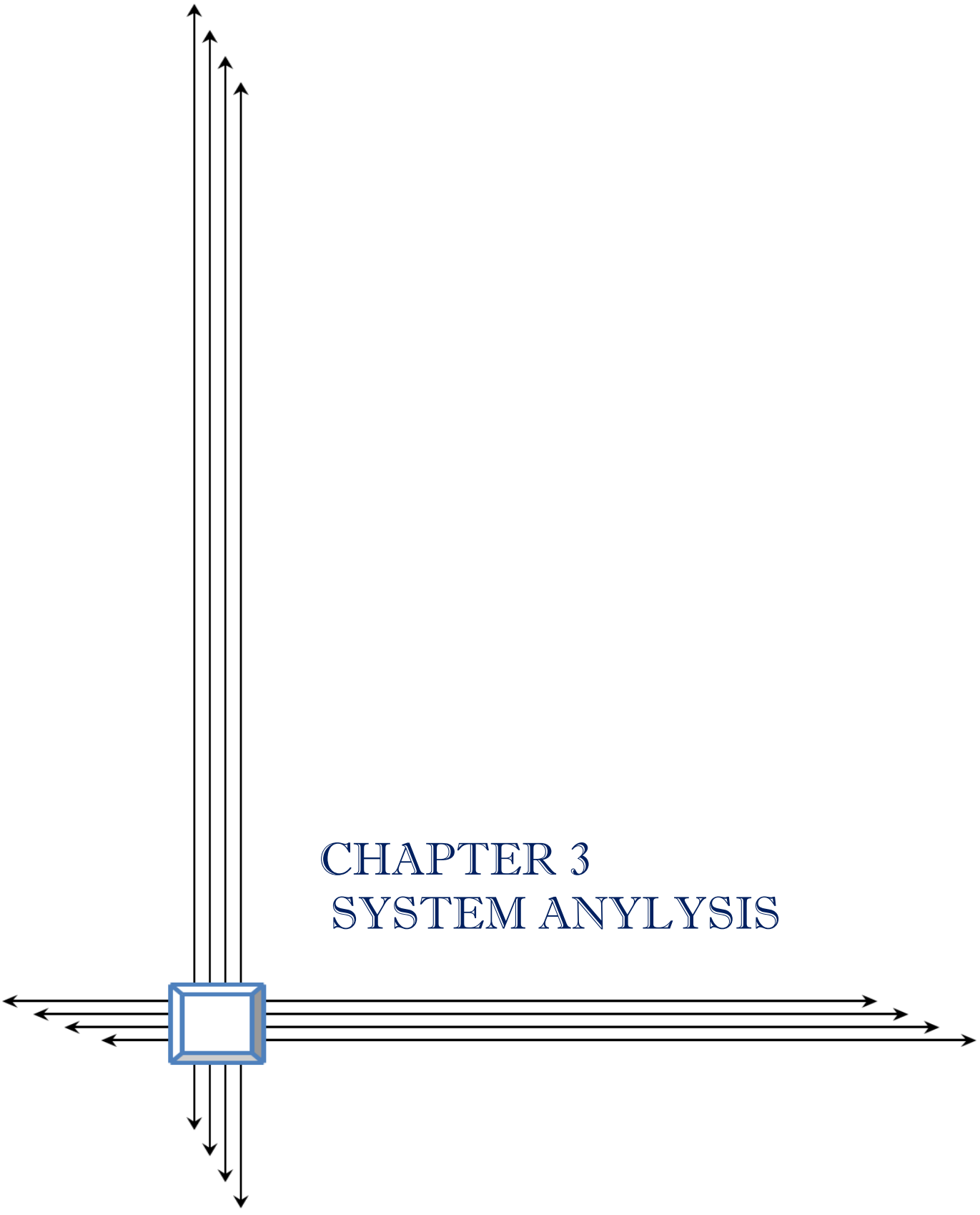
PROBLEM STATEMENT

Usually, user needs to manually manage multiple sets of applications to complete one task. For example, a user trying to check for currently active covid cases for a particular country. In order to achieve this user needs to open a browser and search for a website which displays this particular data and then user needs to enter the country name there then finally user will get all the data which the user is looking for. So there is a need of a system that can manage tasks effortlessly.

There are number of people who have issues in voice recognition. These systems can understand English phrases but they fail to recognize in our accent. Our way of pronunciation is way distinct from theirs. Also, they are easy to use on mobile devices than desktop systems.

There is need of a virtual assistant that can understand English in Indian accent and work on desktop system. When a virtual assistant is not able to answer questions accurately, it's because it lacks the proper context or doesn't understand the intent of the question. Its ability to answer questions relevantly only happens with rigorous optimization, involving both humans and machine learning. Continuously ensuring solid quality control strategies will also help manage the risk of the virtual assistant learning undesired bad behaviors. They require large amount of information to be fed in order to work efficiently.

Virtual assistant should be able to model complex task dependencies and use these models to recommend optimized plans for the user. It needs to be tested for finding optimum paths when a task has multiple sub-tasks and each sub-task can have its own sub-tasks. In such a case here can be multiple solutions to paths, and it should be able to consider user preferences, other active tasks, and priorities in order to recommend a particular plan.



CHAPTER 3
SYSTEM ANYLYSIS

SYSTEM ANALYSIS

Personal assistant software is required to act as an interface into the digital world by understanding user requests or commands and then translating into actions or recommendations based on agent's understanding of the world.

David focuses on relieving voice as primary means of user input. Agent then applies voice recognition algorithms to this input and records the input. It then use this input to call one of the personal information management applications such as task list or calendar to record a new entry or to search about it on search engines like Google. Focus is on capturing the user input through voice, recognizing the input and then executing the tasks if the agent understands the task. Software takes this input in natural language, and so makes it easier for the user to input what he or she desires to be done.

3.1 FEASIBILITY STUDY

Feasibility study can help you determine whether or not you should proceed with your project. It is essential to evaluate cost and benefit. It is essential to evaluate cost and benefit of the proposed system. Five types of feasibility study are taken into consideration.

1. **Technical feasibility:** It includes finding out technologies for the project, both hardware and software. For virtual assistant, user must have microphone to convey their message and a speaker to listen when system speaks. These are very cheap now a days and everyone generally possess them. Besides, system needs internet connection. While using David, make sure you have a steady internet connection. This should not be an issue in this era where almost every home and office has Wi-Fi.

2. **Operational feasibility:** It is the ease and simplicity of operation of proposed system. System does not require any special skill set for users to operate it. In fact, it is designed to be used by almost everyone. Kids who still don't know to write can read out problems for system and get answers.

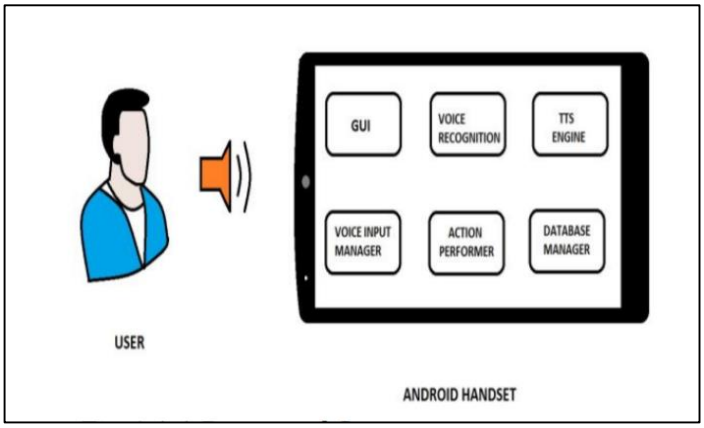
3. **Economical feasibility:** Here, we find the total cost and benefit of the proposed system over current system. For this project, the main cost is documentation cost. As far as maintenance is concerned, David won't cost too much.

4. **Organizational feasibility:** This shows the management and organizational structure of the project. This project is built by a team of two members. The management tasks are all carried out by the team. That won't create any management issues and will increase the feasibility of the project.

5. **Cultural feasibility:** It deals with compatibility of the project with cultural environment. Virtual assistant is built in accordance with the general culture. This is the reason why we named it as David.

3.2 SYSTEM DESIGN

Proposed System: The following figure gives a brief idea about the system architecture. The following figure gives a brief idea about the proposed system.



3.2.1 Main Modules:

Speech Recognition module: The system uses Google's speech recognition system for converting speech input to text. This module uses the library of speechRecognition 3.8.1 for performing speech recognition, with support for several engines and APIs, online and offline.

Python backend: To get the output from the speech recognition module and then identifies whether the command or the speech output is an API Call, Context Extraction, and System Call then the output is send back to the python backend to give the required output to the user.

API Calls: It allows two applications to talk to each other. API is working as a messenger that delivers request to the provider that are requesting it from and then delivers the response back.

Context Extraction: It is the function of automatically or robotic extracting structured information from unstructured or semi-structured or both machine readable documents. In majority of the instance this task concerns processing of human language texts by the method of natural language processing (NLP).

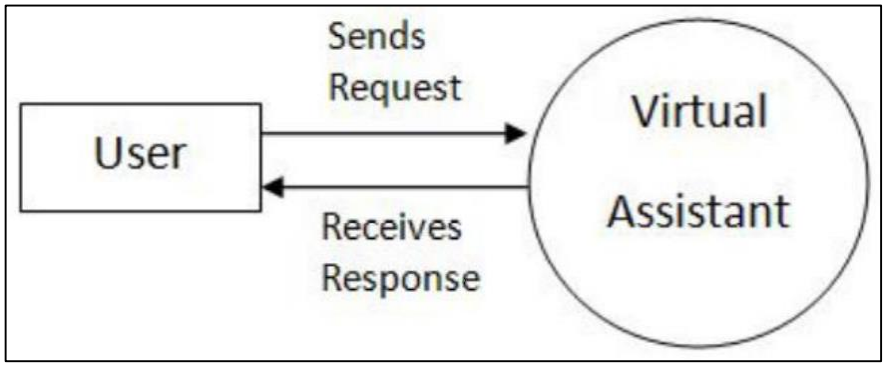
System Calls: It is the programmatic method in which a computer program (function) appeal a service from the kernel of the operating system it is executed on. It includes hardware related assistant and services like access of hard disk drive, creation and execution of new processes, and interacting with integral kernel services like process scheduling. It delivers an essential interface between a process and the operating system.

Text-To-Speech (TTS) Engine: A text-to-speech (TTS) system converts normal language text into speech. Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. The output is given in the form of speech

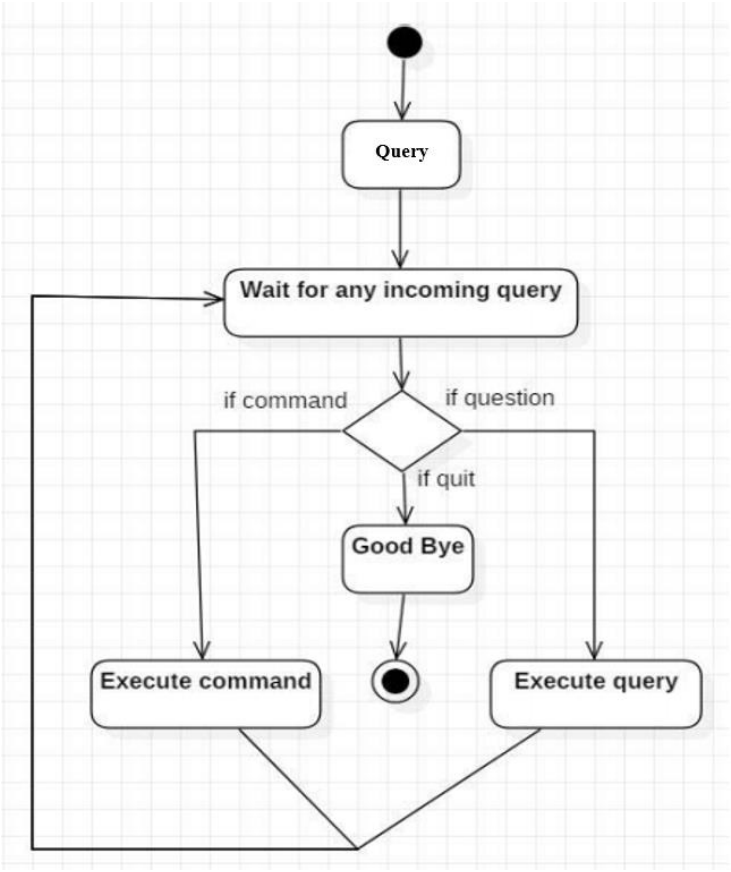
GUI: It is used to interact with the user. GUI reflects the basic appearance of the application.

3.3 Data Flow Diagrams (DFD)

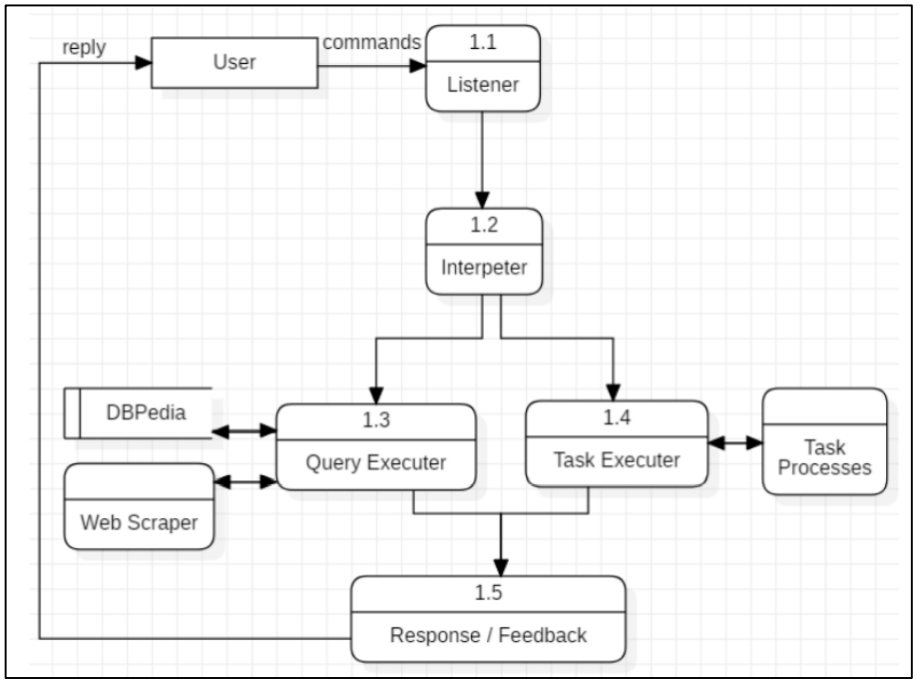
DFD Level 0 :



DFD Level 1 :



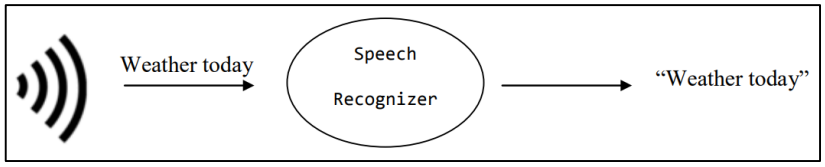
DFD Level 2 :



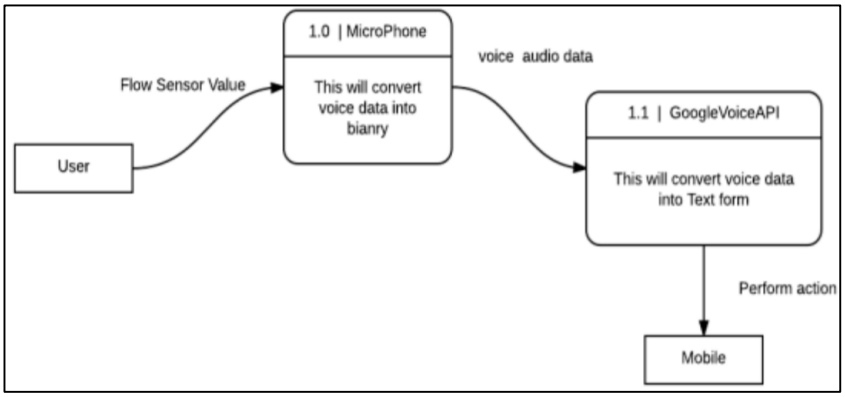
Speech Recognition DFD:

User gives the input into the form of voice, this voice command is recognized by the application. Then action is performed as per the command given. Command given is compared with the database.

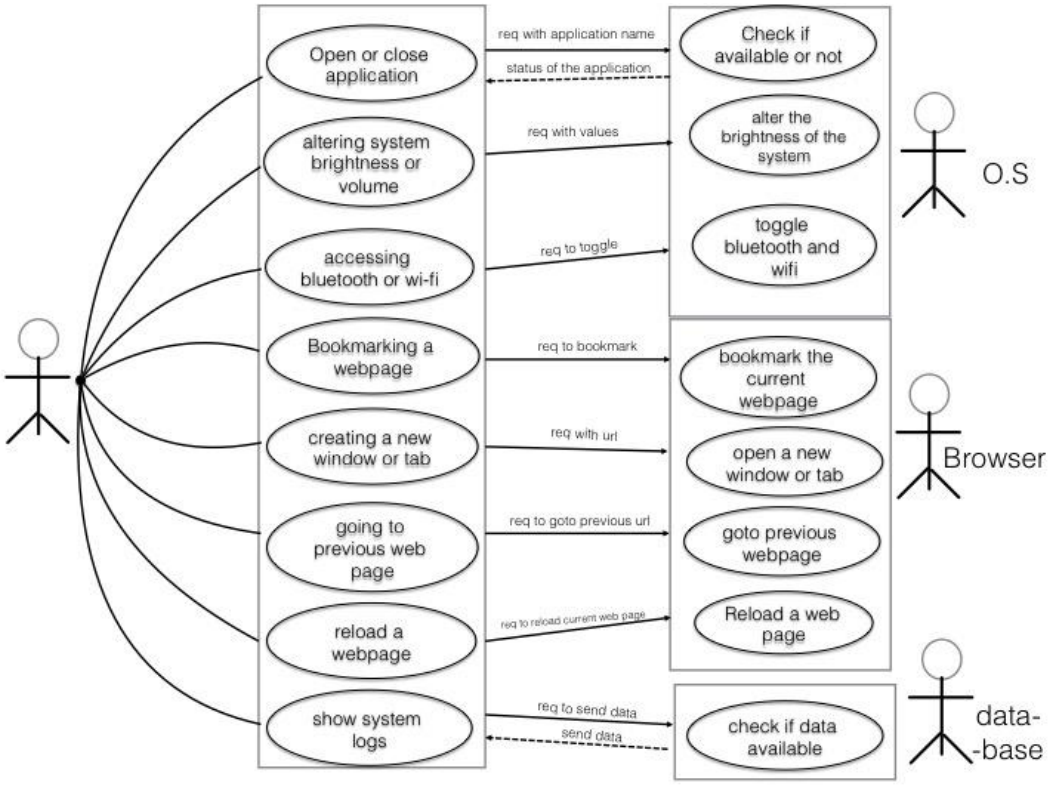
Level 0 :



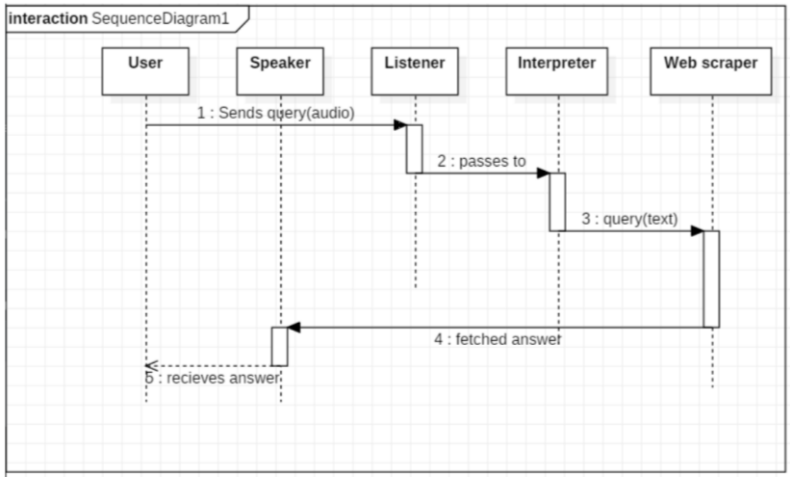
Level 1 :



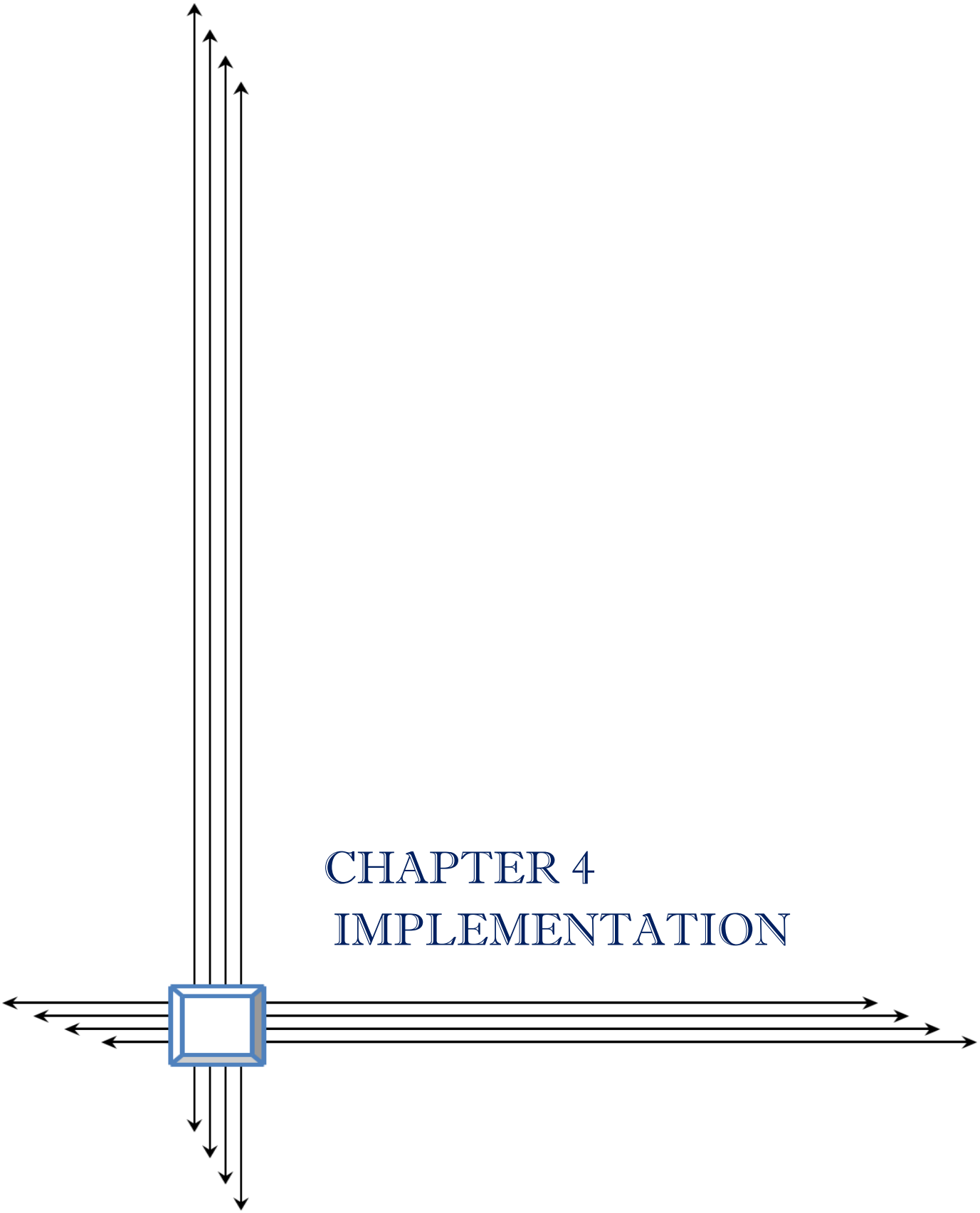
Use Case Diagram:



Sequence Diagram:



The above sequence diagram shows how an answer asked by the user is being fetched from internet. The audio query is interpreted and sent to Web scraper. The web scraper searches and finds the answer. It is then sent back to speaker, where it speaks the answer to user.



CHAPTER 4
IMPLEMENTATION

IMPLEMENTATION

The project aim is to build a personal voice assistant that will make easy for users to use computer with voice command and make task easier. To implement intelligent personal voice assistant, python libraries and Google Speech Recognition API's are used for speechRecognition module and to interpret voice response.

Intelligent personal voice assistants was meant to control the applications online as well as offline and to analyze the performance of the project the working of Intelligent personal voice assistant with desktop application can be opening YouTube with voice command.

The program should firstly be started on the desktop; the initial mode of the program is Voice mode since this program aims at making a voice assistant program. After the program has been started, the user should have correct voice input "command/request" to make those functions work properly.

Input is given by user in the form of voice. Using microphone, voice is converted in binary. GoogleVoiceAPI will convert this voice data in text form and then the action is performed according to the command given by the user by comparing with the match case.

And this program includes the functions and services of: calling services, text message transformation, event handler, services, music player service, checking weather, Google searching engine, Wikipedia searching engine, robot chat, and camera.

Once the user gives a command to the assistant the program first identifies the query and then it converts the query into string after that the assistant processes the query based upon the match cases provided. For example if the user asks to "open Notepad" the main part here is notepad which is an application program and the attached keyword is open thus the match case for the query will be "Notepad" and the operation associated with the notepad which is "open". Finally the assistant initiates a system call to launch a process for opening notepad and the Notepad opens, and assistant gives an acknowledgement to the user that the notepad has been opened.

The details below explain how all functions work and different possibilities while facing different commands.

Calling service: The calling function allows the users to give a call to the person in the contacts. By giving a correct command with the calling request to a stored person, the Assistant will search for the contact name and get the contact of the person, then successfully placing the call the person.

Text Message The calling function allows the users to send a whatsapp message to the person in the contacts. By giving a correct command contains the text or message keyword to send message together with the destination person; the program will navigate to the sending message function on the whatsapp app, message content. The message will be sent to the destination immediately if the user selects to send it with the correct content.

Note taking: The application allows the user to set as many events as they want. Customers set the events with the content and note name, the program switch to the note taking interface with the content and the title, and the event will be stored immediately.

Music player: The music player offers the services to the user to play a named or random song on YouTube depending on the request. The music player service will play the specific song according to the name given by the user. By giving the correct commands, the working music player can stop playing

Checking weather: Weather service provides the user the weather condition in different city. This service works in the same logic and gives back different result depending on the requested city. The weather service return the weather condition of the city with the humidity, wind speed, temperature scope and display in a formalized entity which can be easily read by the user.

Google searching engine: The search engine enable the use to search anything on Google. By detecting the search keyword and search request, the Google search engine will returns the result list displayed to the user through the UI.

Wikipedia searching engine: The search engine enable the use to search anything on Wikipedia. By detecting the Wikipedia keyword and search request, the Wikipedia search engine will returns the Wikipedia result displayed to the user.

General Chat: The assistant exchanges dialog with the user like a normal human enables the user to talk with the Assistant to have fun.

Camera: The camera function enables the user to capture the current view with the camera on the desktop.

Computational problems: The use of certain API's enable the user to ask the assistant for complex solution like solving math operation and asking knowledge based questions like "Who is the prime minister of India " the assistant is smart enough to answer all such questions with accuracy.

Modifying system settings: The assistant is able to change or modify system setting as per user command like changing the brightness level and volume level.

Project Modules:

- **Scraping:**

1. **BeautifulSoup:** Beautiful Soup is a library that makes it easy to scrape information from web pages. It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree.

2. **Wikipedia**: Wikipedia is a Python library that makes it easy to access and parse data from Wikipedia. Search Wikipedia, get article summaries, get data like links and images from a page, and more. Wikipedia wraps the MediaWiki API so you can focus on using Wikipedia data, not getting it.

- **System modules:**

1. **OS:** The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.
2. **Psutil:** It is a Python cross-platform library used to access system details and process utilities. It is used to keep track of various resources utilization in the system. Usage of resources like CPU, memory, disks, network, sensors can be monitored
3. **Pytttsx3:** It is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the pyttsx3.init() factory function to get a reference to a pyttsx3. Engine instance. it is a very easy to use tool which converts the entered text into speech. The pyttsx3 module supports two voices first is female and the second is male which is provided by "sapi5" for windows.
4. **Speech_recognition:** Speech Recognition is an important feature in several applications used such as home automation, artificial intelligence, etc. This article aims to provide an introduction on how to make use of the SpeechRecognition library

of Python. This is useful as it can be used on microcontrollers such as Raspberri Pis with the help of an external microphone.

5. **Tkinter:** Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
6. **Platform:** The Platform module is used to retrieve as much possible information about the platform on which the program is being currently executed. Now by platform info, it means information about the device, it's OS, node, OS version, Python version, etc.
7. **Time:** This module provides various time-related functions. For related functionality, see also the datetime and calendar modules. time module available in Python which provides functions for working with times, and for converting between representations. The function *time.time()* returns the current system time.

- ## Installed modules:

1. **pywhatkit:** PyWhatKit is a Python library for Sending whatsapp message at certain time, it has several other features like Send WhatsApp messages, Play a YouTube video, Perform a Google Search and Get information on particular topic.
2. **covid:** A new Python library which tells the COVID-19 related information (country-wise) and it show that how many cases of confirmed, active, deaths, recovered found in that particular Country.
3. **webbrowser:** webbrowser module provides a high-level interface which allows displaying Web-based documents to users. The webbrowser module can be used to launch a browser in a platform-independent manner. The webbrowser module provides a high-level interface to allow displaying Web-based documents to users.
4. **pyautogui:** PyAutoGUI lets Python control the mouse and keyboard, and other GUI automation tasks. PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.
5. **speedtest:** Speedtest-cli is a module that is used in the command-line interface for testing internet bandwidth using speedtest.net.
6. **PIL:** The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

- **API's**

  1. **WolframAlpha**: The WolframAlpha Web service API provides a web-based API allowing the computational and presentation capabilities of Wolfram Alpha to be integrated into web, mobile, desktop, and enterprise applications. Wolfram Alpha is an API which can compute expert-level answers using Wolfram's algorithms, knowledgebase and AI technology.

- **User defined**

  1. **generalResponse:** This module helps the assistant to answer user's general queries "like how are you" etc. The module contains a dictionary with user query as keys and assistant response as value to the key.
  2. **game:** This module is a Tic tac toe module which will be used when user queries for playing game.
  3. **YoututbeDownloader:** This module is for downloading YouTube videos based upon the provided link by the user. This is an extension to a module named pytube.

Some Usage Commands:

1. "Play a song for me", the program will ask which song you want to play.
2. "I want to call", make a phone call to Lucy. The program will capture the command keyword "call" and then it will ask for the contact name.
3. "Send a message", the program will capture the keyword "message" and then it will ask for the name of the contact along with the text you want to send.
4. "Set up a meeting at 10", the Program will first capture the keyword "meeting" then the event activity will be start with the add event dialog, automatically creating a meeting link.
5. "Google Spain", the keyword 'Google' is detected and the result will be presented on the web browser by searching 'Spain" on Google.
6. "Open the camera", as the keyword 'open camera' is detected, the camera is started. And the program will take a photo within 5 seconds.

HARDWARE AND SOFTWARE REQUIREMENTS

The software is designed to be light-weighted so that it doesn't much resources of the machine and eventually be a burden on the machine running it. This system is being build keeping in mind the generally available hardware and software compatibility. Here are the minimum hardware and software requirement for virtual assistant.

RAM 512MB or more. Software: • Windows 7(32-bit) or above. • Python 2.7 or later • Chrome Driver • Selenium Web Automation • SQLite

Software Requirements:

• Windows 7(32-bit) or above.
• Python 3.9.1 or later
• Pycharm Community Edition 2020.3

Hardware Requirements:

- Ram: 512Mb minimum, 1GB Recommend.
- Storage: 2GB Recommend.
- CPU: Core 2 Duo Minimum, I3 5th Gen Recommend

Functional Requirements:

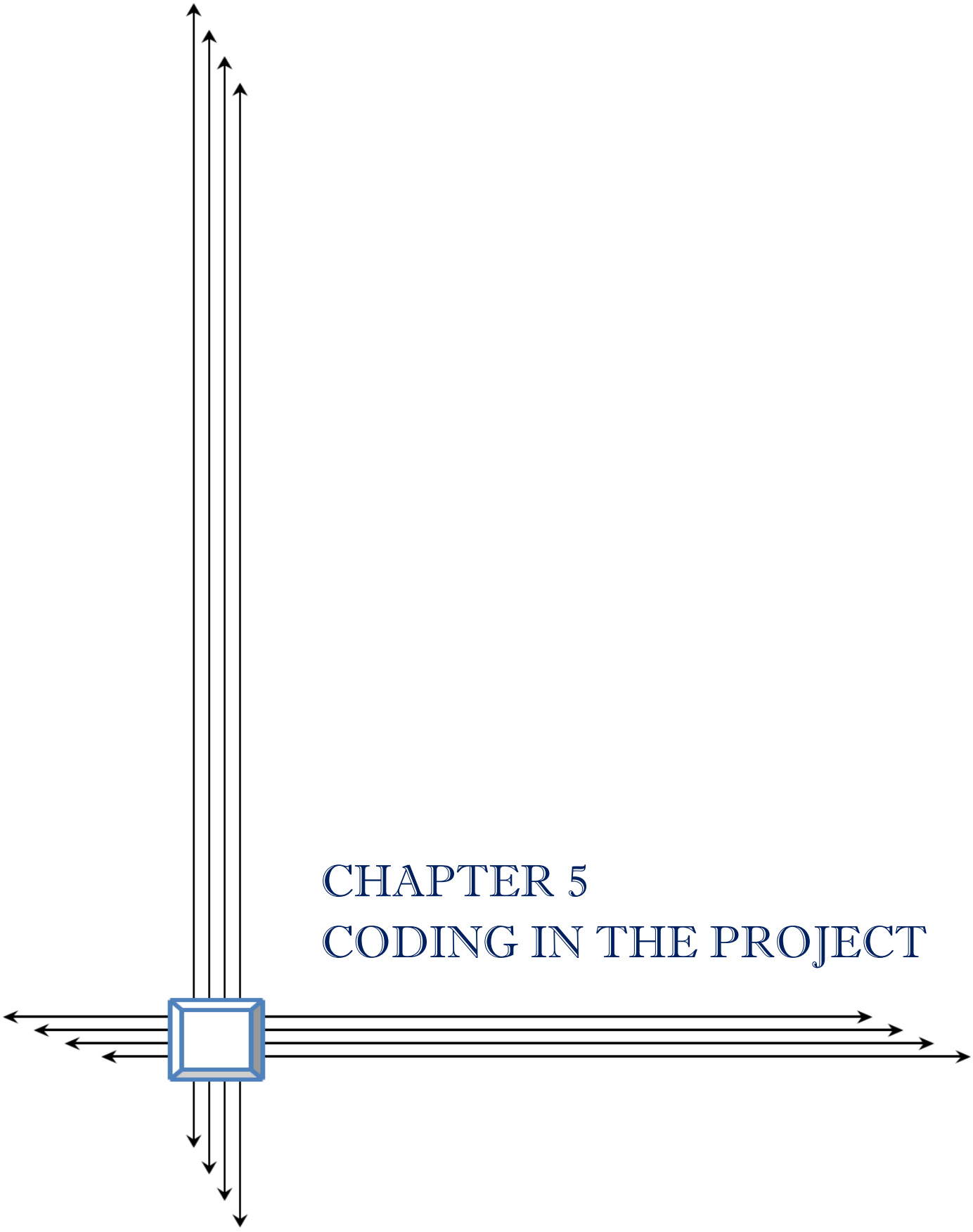
The main purpose of functional requirements within the requirement specification document is to define all the activities or operations that take place in the system. These are derived through interactions with the users of the system.

1. Accepting an audio signal,
2. Deciding whether the audio signal is addressed to the software agent,
3. Attempting to map a relevant signal onto a word sequence,
4. Determining the rover action request associated with the understood word sequence,
5. Providing appropriate feedback to the user.

Non-functional Requirements:

Non-functional requirements describe user-visible aspects of the system that are not directly related to functionality of the system.

- The system ensures safety, security and usability, which are observable during operation (at run time).
- The system is adaptable to different situations.
- The project has good and compact UI using Tkinter with responsive interface.
- The project is light on resources.



CHAPTER 5
CODING IN THE PROJECT

CODING:

- **Main**

```
# importing all the required modules
import pyttsx3
import pywhatkit=
import speech_recognition as sr
import psutil
import cv2
from covid import *=
from requests import get
import wikipedia
import webbrowser=
import pywhatkit as kit
import pyjokes
import pyautogui=
from pyautogui import *
from keyboard import *
from time import sleep
from generalResponse import responseDictionary
import os
import screen_brightness_control as sbc
import speedtest=
import requests
import platform
import wolframalpha
from bs4 import BeautifulSoup
from tkinter import *
from PIL import Image, ImageTk=

# initialising speech engine
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[2].id)
engine.setProperty('rate', 175)

queryLogger = []
queryIndex = -1

window = Tk()
window.geometry('800x800')
window.minsize(630, 395)
window.maxsize(630, 395)

var = StringVar()
var1 = StringVar()

labelImage = Image.open('images/labelBackground.jpg')
resizedLabelImage = labelImage.resize((200, 50), Image.ANTIALIAS)
updatedLabelImage = ImageTk.PhotoImage(resizedLabelImage)

startButtonImage = Image.open('images/Start.png')
resizedStartImage = startButtonImage.resize((125, 40), Image.ANTIALIAS)
updatedStartButtonImage = ImageTk.PhotoImage(resizedStartImage)
```

```
ExitButtonImage = Image.open('images/Quit.png')
resizedExitImage = ExitButtonImage.resize((125, 40), Image.ANTIALIAS)
updatedExitButtonImage = ImageTk.PhotoImage(resizedExitImage)

backgroundImage = Image.open('images/back1.png')
resizedBackgroundImage = backgroundImage.resize((630, 395),
Image.ANTIALIAS)
updatedBackgroundImage = ImageTk.PhotoImage(resizedBackgroundImage)

# speak function for text to speech output
def speak(audio):
    engine.say(audio)
    var.set(audio)
    window.update()
    print(audio)
    engine.runAndWait()

query = ''

# this allows the program to take user input in the form of audio //speech
recognition functionality
# noinspection PyBroadException
def takecommand():
    global query, queryIndex, queryIndex

    r = sr.Recognizer()
    with sr.Microphone() as source:
        print('listening...')
        var1.set('listening...')
        window.update()
        r.pause_threshold = 1
        audio = r.listen(source, timeout=4, phrase_time_limit=7)

    try:
        print('recognising...')
        var1.set('recognising...')
        window.update()
        query = r.recognize_google(audio, language='en-in')
        print(f'user said: {query}')
        var1.set(query)
        window.update()
        queryLogger.append(query)
        queryIndex += 1

    except Exception:
        query = ''
    return query.lower()

# initialising system
def initialiseSystem():
    try:
        speak("Initialising system. Hello, i am David, your Personal Voice
Assistant,"
              " from now i will be your companion in most of the tasks, and
i will assist you as best as i can.")
        ramUsage = psutil.virtual_memory()[2]
```

```
            cpuUsage = psutil.cpu_percent(2)
            batteryCheck = psutil.sensors_battery()
            percentage = batteryCheck.percent
            speak(f"Your current battery level is at {percentage} %")
            speak(f'RAM memory {ramUsage} % used:')
            speak(f'The CPU usage is: {cpuUsage} %')
            speak("All systems are fully operational.")
            speak("Now tell me, what can i do for you?")
        except:
            speak("Some error has occurred, wait for a moment i am fixing it")

    # greets the user along with time
    def wish():
        try:
            hour = int(datetime.datetime.now().hour)
            tt = time.strftime("%I:%M %p")

            if hour in range(0, 13):
                speak(f'Good Morning, its {tt}')
            elif hour in range(12, 18):
                speak(f'Good Afternoon, its {tt}')
            else:
                speak(f'Good Evening, its {tt}')
        except:
            speak("Some error has occurred")

    # key board shortcut function
    def keyboardShortcut(shortcutQuery):
        try:
            file = open('File.txt', 'r')
            value = ''
            counter = 0
            keyResults = shortcutQuery.lower()
            keyResults = keyResults.split()
            if 'of' in keyResults:
                searchIndex = keyResults.index('of')
                value = keyResults[searchIndex + 1]
                value = value.capitalize()
            if 'for' in keyResults:
                searchIndex = keyResults.index('for')
                value = keyResults[searchIndex + 1]
                value = value.capitalize()
            for line in file:
                if value in line:
                    counter += 1
                    speak(line)
                    if counter == 3:
                        break
            if counter == 0:
                speak("Sorry i couldn't find any relevant solution for this")
            else:
                speak(f' i found {counter} solutions ')
        except:
            speak("It seems like, some system files are missing please check
    all program related files.")
```

```
# opens google chrome with google as home page
def launchGoogle():
    try:
        speak("sir, what should i search on google")
        searchQuery = takecommand().lower()
        if "i will" in searchQuery or "not" in searchQuery or "don't" in
searchQuery:
            webbrowser.open("https://www.google.com")
        else:
            webbrowser.open("https://www.google.com/search?q=" +
searchQuery)
    except:
        speak("Something went wrong while performing search")

# this function returns currently active cases in india
def covidCases():
    try:
        currentCases = Covid()
        speak("Which country data you are looking for")
        countryName = takecommand().capitalize()
        case = currentCases.get_status_by_country_name(countryName)
        country = case['country']
        active = case['active']
        deaths = case['deaths']
        recovered = case['recovered']
        speak(f"Country is {country}\n"
              f"Currently active cases are {active}\n"
              f"Total recovered cases are {recovered}\n"
              f"Total deaths till now are {deaths}\n")
    except:
        speak("that doesn't seem to be a country name, please try again")
        covidCases()

# whatsapp automation
def whatsappAutomation(name, message):
    speak('Opening whatsapp')

os.startfile('C:\\Users\\LENOVO\\AppData\\Local\\WhatsApp\\WhatsApp.exe')
    sleep(8)
    click(325, 142)  # search contact
    sleep(3)
    write(name)  # search name
    sleep(3)
    click(305, 295)  # click on search result
    sleep(3)
    click(990, 990)  # click on message input field
    sleep(3)
    write(message)   # write message in enter message field
    sleep(1)
    press('enter')  # hit enter
    speak(f'Your message has been sent successfully to {name}')

# whatsapp voice call
def whatsappCall(calltype, name):
    speak('Opening whatsapp')
```

```
os.startfile('C:\\Users\\LENOVO\\AppData\\Local\\WhatsApp\\WhatsApp.exe')
    sleep(8)
    click(325, 142)  # search contact
    sleep(2)
    write(name)  # search name
    sleep(3)
    click(305, 295)  # click on search result
    sleep(3)
    if calltype == 'voice':
        click(1724, 73)  # click on call button
    elif calltype == 'video':
        click(1661, 73)  # click on video call button
    speak(f"Calling, {name}")

# creating meeting link
def createMeetLink():
    speak("Opening google meet to start an instant meeting")
    webbrowser.open("https://meet.google.com/")
    sleep(5)
    click(205, 712)
    sleep(5)
    click(285, 768)
    speak("Your meeting has started, now you can ask others to join your
meeting")

# joining pwp class
def joinClass(code):
    speak("Opening google meet to join class")
    webbrowser.open("https://meet.google.com/")
    sleep(10)
    click(510, 720)      # entering the class code
    sleep(2)
    write(code)      # paste class code
    sleep(2)
    click(643, 717)      # click on join
    sleep(10)
    click(1305, 638)     # click on ask to join
    speak("Wait until someone lets you in")

# changing system volume as per user convenience
def changeVolume(string):
    try:
        getString = ''
        if 'change' in string:
            speak("by how much % should i increase or decrease the
volume?")
            getString = takecommand()
            getValue = getIntegers(getString)
        else:
            getValue = getIntegers(string)
        taps = int(getValue / 10) * 5
        if 'up' in string or "increase" in string or 'up' in getString or
"increase" in getString:
            for i in range(taps):
                pyautogui.press('volumeup')
```

```
            speak(f"Volume level increased by {getValue}%")
        elif 'down' in string or 'decrease' in string or "lower" in string
or 'down' in getString or 'decrease' in getString or "lower" in getString:
            for i in range(taps):
                pyautogui.press('volumedown')
            speak(f"Volume level decreased by {getValue}%")

    except:
        speak("Something went wrong, while changing system volume")

# news function
def NewsFromBBC():
    # BBC news api
    # following query parameters are used
    # source, sortBy and apiKey
    query_params = {
        "source": "bbc-news",
        "sortBy": "top",
        "apiKey": "4dbc17e007ab436fb66416009dfb59a8"
    }
    main_url = " https://newsapi.org/v1/articles"

    # fetching data in json format
    res = requests.get(main_url, params=query_params)
    open_bbc_page = res.json()

    # getting all articles in a string article
    article = open_bbc_page["articles"]

    # empty list which will contain all trending news
    results = []

    for ar in article:
        results.append(ar["title"])

    for i in range(len(results)):
        # printing all trending news
        speak(results[i])

# returns integers only in a given string
def getIntegers(string):
    try:
        if '%' in string:
            string = string.replace('%', '')
        if '°C' in string:
            string = string.replace('°C', '')
        numbers = [int(word) for word in string.split() if
word.isnumeric()]
        return numbers[0]
    except:
        return []

# increases and decreases brightness as per user command
def brightnessControl(changeBrightnessCommand):
    try:
        brightnessLevel = getIntegers(changeBrightnessCommand)
```

```
            getCurrentBrightness = sbc.get_brightness()
            if "increase" in changeBrightnessCommand:
                sbc.set_brightness(getCurrentBrightness + brightnessLevel)
                speak(f"Brightness increased by {brightnessLevel}%")

            else:
                sbc.set_brightness(getCurrentBrightness - brightnessLevel)
                speak(f"Brightness decreased by {brightnessLevel}%")
        except:
            speak("something went wrong while changing brightness level")

    # this function returns day of week
    def getDay():
        day = datetime.datetime.today().weekday()+1
        print(day)
        dayDict = {1: "Monday", 2: "Tuesday", 3: "Wednesday", 4: "Thursday", 5:
    "Friday", 6: "Saturday", 7: "Sunday"}
        if day in dayDict.keys():
            speak(f"it's {dayDict[day]} today")

    # to check for internet speed
    def getInternetSpeed():
        speak("Checking for your internet speed, this might take a while")
        speed = speedtest.Speedtest()
        upload = round(speed.upload() / 10 ** 6, 2)
        download = round(speed.download() / 10 ** 6, 2)
        speak(f"Your upload speed is {upload} MBPS \nYour download speed is
    {download} MBPS ")
        if upload < 5.0:
            speak("Your upload speed is slow")
        if download < 10:
            speak("Your download speed is slow")
        if download in range(10, 16):
            speak("Your download speed is average")
        if download > 20:
            speak("Your download speed is perfectly fine")

    # system information function
    def systemInformation():
        my_system = platform.uname()
        speak(f"Your system is {my_system.system}\n"
              f"Node name is {my_system.node}\n"
              f"Release {my_system.release}\n"
              f"Version {my_system.version}\n"
              f"Machine {my_system.machine}\n"
              f"Processor {my_system.processor}")

    # main function in which main logic of code resides
    def runAssistant():
        global query
        wish()
        #initialiseSystem()
        while True:
            query = takecommand()
            if "notepad" in query or 'editor' in query:
```

```
            if "open" in query:
                speak("Opening Notepad")
                os.startfile("C:\\Windows\\System32\\notepad.exe")
            elif "close" in query:
                speak("Notepad closed")
                os.system("taskkill /f /im notepad.exe")
            continue

        # for command prompt
        elif "command prompt" in query or 'cmd' in query or 'terminal' in
query:
            if "open" in query:
                speak("Opening Command prompt ")
                os.system("start cmd")
            elif "close" in query:
                speak("Command prompt terminated")
                os.system("taskkill /f /im cmd.exe")
            continue

        #  to open camera
        elif "open camera" in query:
            speak("Opening Camera, get ready")
            cap = cv2.VideoCapture(0)
            ret, img = cap.read()
            cv2.imshow('webcam', img)
            k = cv2.waitKey(10)
            if k == 27:
                cap.release()
                cv2.destroyAllWindows()
            continue

        # play music

        elif "music" in query or "song" in query:
            speak("sir, what song should i play")
            searchQuery = takecommand().lower()
            if 'play' in searchQuery:
                searchQuery = searchQuery.replace('play', '')
            kit.playonyt(searchQuery)
            speak("your song is being played on youtube")
            speak("Sir i am still here in case you need me.")
            continue

        # for ip address
        elif "ip address" in query:
            ip = get("https://api.ipify.org").text
            speak(f"your ip address is {ip}")
            continue

        # wikipedia
        elif "wikipedia" in query:
            try:
                speak("What should i search on wikipedia")
                cm = takecommand()
                webbrowser.open("https://en.m.wikipedia.org/wiki/" + cm)
                results = wikipedia.summary(cm, 2)
```

```
                speak("according to wikipedia" + results)
            except:
                speak("Some error has occurred, try something else")
            continue

        # opening youtube in default browser
        elif "youtube" in query and "open" in query:
            try:
                speak('What should i search on youtube?')
                searchQuery = takecommand()
                if "i will" in searchQuery or "nothing" in searchQuery or
"don't search" in searchQuery:
                    speak("Alright sir, opening youtube")
                    webbrowser.open("www.youtube.com")
                elif 'search for' in searchQuery:
                    searchQuery = searchQuery.replace('search for', '')
                    speak(f"Alright sir, searching for {searchQuery} on
youtube")
                    webbrowser.open("https://www.youtube.com/search?q=" +
searchQuery)
                    time.sleep(5)
                    speak("here are your results. Sir i am still here in
case you need me.")
                else:
                    speak(f"Alright sir, searching for {searchQuery} on
youtube")
                    webbrowser.open("https://www.youtube.com/search?q=" +
searchQuery)
                    time.sleep(5)
                    speak("here are your results. Sir i am still here in
case you need me.")
            except:
                speak("Some error has occurred, try something else")
            continue

        # closing youtube
        elif "close" in query and "youtube" in query:
            press_and_release("ctrl+w")
            speak('Closing Youtube')
            continue

        # opening facebook in chrome
        elif "facebook" in query:
            if "open" in query or "start" in query or 'check' in query:
                speak("Opening your facebook account")
                webbrowser.open("www.facebook.com")
            elif 'close' in query:
                speak("Done, facebook closed")
                press_and_release("ctrl + w")
            speak("Alright sir, i am still here in case you need me.")
            continue

        # opening default browser //chrome
        elif "chrome" in query:
            if "open" in query:
                launchGoogle()
```

```
            elif "close" in query or 'terminate' in query:
                os.system("taskkill /im chrome.exe /f")
                speak("Chrome terminated")
            continue

        # to find joke
        elif "tell me a joke" in query:
            joke = pyjokes.get_joke()
            speak(joke)
            continue

        # google search using web scraping
        # this reads the top article and also redirects the user to the
browser
        elif "google" in query or "search" in query or "tell me about" in
query:
            import wikipedia as googleScrap
            if "google" in query:
                query = query.replace('google', "")
            if "search" in query:
                query = query.replace("search", '')
            elif 'tell me about' in query:
                query = query.replace("tell me about", '')
            else:
                query = query.replace("tell me", '')
            speak("this is what i have found on web")
            pywhatkit.search(query)
            try:
                searchResult = googleScrap.summary(query, 2)
                speak(searchResult)
            except:
                speak("sorry but no data available for your search")
            finally:
                speak("Now you can continue")
            continue

        # switch current window
        elif "switch" in query:
            if "tab" in query or "window" in query:
                pyautogui.keyDown("alt")
                pyautogui.press("tab")
                time.sleep(1)
                pyautogui.keyUp("alt")
                speak("Tab switched")
            continue

        # takes screenshot of the current screen within 1 second
        elif "screenshot" in query:
            myScreenshot = pyautogui.screenshot()

myScreenshot.save(r'C:\Users\LENOVO\Pictures\Screenshots\screenshot1.png')
            speak("Screenshot captured")
            continue

        # tells us the temperature of particular location
        elif "temperature" in query:
```

```
            try:
                query = query.split()
                ind = query.index('in') + 1
                search = "temperature in " + query[ind]
                url = f"https://www.google.com/search?q={search}"
                r = requests.get(url)
                data = BeautifulSoup(r.text, "html.parser")
                temp = data.find("div", class_="BNeawe").text
                try:
                    if int(getIntegers(temp)):
                        speak(f"current {search} is {temp}")
                except:
                    speak("City not found, please check the city name and
try again")
            except:
                    speak("City not found, please check the city name and
try again")
            continue

        # helps in finding keyboard shortcuts
        elif 'shortcut' in query:
            keyboardShortcut(query)
            continue

        # appreciating assistant
        elif "good job" in query or "well done" in query or "thank" in
query or "great job" in query or 'nice' in query:
            speak("That's so nice to hear from you. I am still here in case
you need me")
            continue

        # to find deals and for online shopping
        elif "deal" in query or "shop" in query or "amazon" in query:
            speak("okay, let me open an online shopping site")
            speak("i can open amazon, so i am opening amazon")
            webbrowser.open("https://www.amazon.com/")
            continue

        # opens instagram on user command
        elif "instagram" in query:
            if "open" in query or "start" in query:
                speak("okay, let me open your insta account")
                time.sleep(0.5)
                webbrowser.open("https://www.instagram.com")
            elif 'close' in query:
                press_and_release("ctrl + w")
                speak("Done, instagram closed")
            else:
                speak("Sir, i am still here in case you need me.")
            continue

        # texting on whatsapp
        elif 'message' in query or 'text' in query:
            #if 'whatsapp' in query:
            speak('to whom do you wanna text')
            getContactName = takecommand()
```

```
            speak(f'Tell me what message you want to send to
{getContactName}')
            getUserMessage = takecommand()
            whatsappAutomation(getContactName, getUserMessage)
            speak("Sir, i am still here in case you need me.")
            continue

        # calling on whatsapp
        elif 'call' in query:
            speak("To whom do you wanna call")
            getContactName = takecommand()
            if 'video' in query:
                whatsappCall('video', getContactName)
            else:
                whatsappCall('voice', getContactName)
            continue

        # creating meeting link
        elif 'create meet' in query or 'meet' in query or 'create link' in
query:
            createMeetLink()
            continue

        elif 'join' in query:
           if 'pwp' in query or "python" in query:
               meetingCode = 'uwz-ckdt-aut'
               joinClass(meetingCode)
           continue

        # repeats user query
        elif "repeat" in query or 'respond' in query:
            if "after me" in query or "my word" in query or "what i say" in
query:
                speak("Sure sir, go ahead")
                jj = takecommand()
                speak(f"you said :{jj}")
            continue

        # for saving user notes
        elif "note" in query:
            if "take" in query or "write" in query or 'save' in query:
                speak("What should i write in the note, sir")
                note = takecommand()
                speak("Please select a file name")
                fileName = takecommand()
                file = open(f'{fileName}.txt', 'w')
                speak("Sir, Should i include date and time")
                snfm = takecommand()
                if 'yes' in snfm or 'sure' in snfm:
                    strTime = datetime.datetime.now().strftime("%H:%M:%S")
                    file.write(strTime)
                    file.write(" :- ")
                    file.write(note)
                    speak(f"file saved as {fileName}")
                else:
                    file.write(note)
```

```
                    speak(f"file saved as {fileName}")
            if 'show' in query:
                try:
                    speak("Showing Notes")
                    file = open(f"{fileName}.txt", "r")
                    speak(file.read())
                except:
                    speak("Sorry, but it looks like you don't have any
saved notes yet")
            continue

        # to return latest news headlines
        elif "news" in query or 'headline' in query:
            NewsFromBBC()
            continue

        # tells us system information
        elif "system" in query or "tell me about my system" in query:
            if "info" in query or "about" in query or "status" in query or
'configuration' in query:
                systemInformation()
            continue

        elif "volume" in query:
            changeVolume(query)
            continue

        # opens word
        elif "word" in query:
            if "open" in query or 'start' in query:
                speak("Opening Microsoft word")
                os.startfile("Winword.exe")
            elif 'close' in query:
                speak("Microsoft Word terminated")
                os.system("taskkill /f /im Winword.exe")
            continue

        # opens excel
        elif "excel" in query:
            if "open" in query or 'start' in query:
                speak("Opening Microsoft excel")
                os.startfile("Excel.exe")
            elif 'close' in query:
                speak("Microsoft Excel terminated")
                os.system("taskkill /f /im Excel.exe")
            continue

        # opens powerpoint
        elif "power point" in query or "powerpoint" in query or
"presentation" in query:
            if "open" in query or 'start' in query:
                speak("Opening Microsoft Power Point")
                os.startfile("Powerpnt.exe")
            elif 'close' in query:
                speak("Microsoft Power Point terminated")
                os.system("taskkill /f /im Powerpnt.exe")
```

```
            continue

        elif 'which day' in query:
            getDay()
            continue

        # enables sleep mode for specified time limit
        elif "don't listen" in query or "stop listening" in query:
            speak("for how much time you want me to stop listening for
commands")
            getSleepTime = takecommand()
            if 'minute' in getSleepTime:
                varTimeforSleep = getIntegers(getSleepTime)*60
                if varTimeforSleep == 60:
                    speak(f"sleep mode initiated for 1 minute")
                    time.sleep(varTimeforSleep)
                else:
                    speak(f"sleep mode initiated for
{getIntegers(getSleepTime)} minutes")
                    time.sleep(varTimeforSleep)
            elif 'second' in getSleepTime:
                varTimeforSleep = getIntegers(getSleepTime)
                speak(f"sleep mode initiated for {varTimeforSleep}
seconds")
                time.sleep(varTimeforSleep)
            elif 'hour' in query:
                speak("sorry, but i cannot stay active in background for
that much long time")
            else:
                speak(f"sleep mode initiated for default 1 minute")
                time.sleep(60)
            continue

        # downloads youtube video
        elif "download" and "youtube" in query:
            speak("Sir please provide the video link and select the
destination in order to download the video")
            os.system("python YoutubeDownloader.py")
            speak("Your Video is downloaded")
            continue

        # tells us covid cases in a particular country
        elif "covid" in query or "corona" in query or 'pandemic status' in
query or 'pandemic situation' in query:
            covidCases()
            continue

        # to open pycharm
        elif 'code' in query or 'update' in query or 'project' in query:
            os.startfile("C:\\Program Files\\JetBrains\\PyCharm Community
Edition 2020.3\\bin\pycharm64.exe")
            speak("Alright sir, opening py charm for you.")
            sleep(2)
            speak("This might take a while")
            continue
```

```
        elif "do nothing" in query:
            speak("Alright sir, i am still here in case you need me")
            continue

        elif 'initiate' in query or 'initialise' in query or 'reboot' in
query:
            initialiseSystem()
            continue

        # this part reminds user his/her message
        elif "remember" in query:
            remember = open("remember.txt", "w")
            if 'remember that' in query:
                rememberMsg = query.replace("remember that", "")
                speak("you asked me to remind you that :" + rememberMsg)
                remember.write(rememberMsg)
            elif 'do you remember' in query:
                rememberFile = open("remember.txt", "r")
                speak("you asked me that" + rememberFile.read())
            else:
                speak("What do you want me to remember?")
                rememberText = takecommand()
                if 'remember that' in rememberText:
                    rememberMsg = rememberText.replace("remember that", "")
                    speak("you asked me to remember that :" + rememberMsg)
                    remember.write(rememberMsg)
                else:
                    speak("you asked me to remember that :" + rememberText)
                    remember.write(rememberText)
            remember.close()
            continue

        elif 'bore' in query or 'game' in query or 'play' in query:
            if 'bore' in query:
                speak("in that case, i can help to get rid of boredom. Do
you wanna play tic tac toe? or you want to listen a joke")
                gameName = takecommand()
                if 'game' in gameName or "tic" in gameName or 'play' in
gameName:
                    speak('Alright, get ready to play tic tac toe')
                    os.system('python Game.py')
                if 'joke' in gameName:
                    joke = pyjokes.get_joke()
                    speak(joke)
            elif 'game' in query or 'play' in query:
                speak('Alright, get ready to play tic tac toe')
                os.system('python Game.py')
            continue

        # changing brightness level
        elif 'bright' in query:
            brightnessControl(query)
            continue

        elif 'sleep' in query:
```

```
            speak("yeah, i also feel that i should take a nap, let me have
a quick power nap of 5 minutes.")
            sleep(300)
            continue

        # to check internet speed
        elif 'speed' in query or 'connection' in query or "internet" in
query:
            getInternetSpeed()

        # for waiting for a certain amount of time
        elif 'wait' in query or 'hold' in query:
            varTimeforSleep = getIntegers(query)      # this will return a
list of numbers in query
            if 'minute' in query:
                varTimeforSleep *= 60
                if varTimeforSleep == 60:
                    speak(f"waiting for 1 minute till the next command")
                    sleep(varTimeforSleep)
                else:
                    speak(f"waiting for {getIntegers(query)} minutes till
the next command")
                    time.sleep(varTimeforSleep)
            elif 'second' in query:
                speak(f"waiting for {varTimeforSleep} seconds till the next
command")
                time.sleep(varTimeforSleep)
            speak("Tell me sir, what you want me to do now?")
            continue

        #for responding to general questions
        elif query in responseDictionary.keys():
            user = query
            dictVar = responseDictionary
            if user in dictVar.keys():
                speak(dictVar[user])
            continue

        elif "rest" in query or 'bye' in query or 'close project' in query
or "shut" in query:
            if "shut" in query:
                speak("okay sir. Terminating all programs make sure you
have saved all your files.")
            else:
                speak("Okay Sir, you can call me anytime. Now let me have
some rest.")
            sys.exit()

        # final block for wolframalpha
        else:
            try:
                question = query
                app_id = '4E8LLP-TX5UPAUYVP'
                client = wolframalpha.Client(app_id)
                res = client.query(question)
                answer = next(res.results).text
```

```
                answer = answer.splitlines()
                if len(answer) < 2:
                    speak(answer[0])
                if len(answer) > 2:
                    speak(answer[0])
                    speak(answer[1])
            except:
                speak("can you please say that again")
            continue

def exitAssistant():
    sys.exit()

# Call main function
#this is the main part where execution starts
if __name__ == "__main__":

    bglabel = Label(window, image=updatedBackgroundImage)
    bglabel.place(x=0, y=0)

    label2 = Message(window, textvariable=var1, width=450, fg='#0f0',
bg='#000', font=('Courier', 10))
    var1.set('User Said:')
    label2.place(x=75, y=75)

    label1 = Message(window, textvariable=var, width=450, fg='#00ff00',
bg='#000', font=('Courier', 10), justify='center')
    var.set('Welcome')
    label1.place(x=75, y=150)

    btn1 = Button(text='Start', bg='#000', command=runAssistant,
image=updatedStartButtonImage)
    btn1.config(font=("Courier", 12))
    btn1.place(x=75, y=275)

    btn2 = Button(text='EXIT', bg='#000', command=exitAssistant,
image=updatedExitButtonImage)
    btn2.config(font=("Courier", 12))
    btn2.place(x=425, y=275)

    window.title('David')
    window.mainloop()
```

- **Game**

```
# Tic Tac Toe game with GUI
# using tkinter
# importing all necessary libraries
import random
import pyttsx3
from tkinter import *
from functools import partial
from tkinter import messagebox
from copy import deepcopy

# sign variable to decide the turn of which player
sign = 0
result=''
# Creates an empty board
global board
board = [[" " for x in range(3)] for y in range(3)]

engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[2].id)
engine.setProperty('rate', 175)

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

# Check l(O/X) won the match or not
# according to the rules of the game
def winner(b, l):
    return ((b[0][0] == l and b[0][1] == l and b[0][2] == l) or
            (b[1][0] == l and b[1][1] == l and b[1][2] == l) or
            (b[2][0] == l and b[2][1] == l and b[2][2] == l) or
            (b[0][0] == l and b[1][0] == l and b[2][0] == l) or
            (b[0][1] == l and b[1][1] == l and b[2][1] == l) or
            (b[0][2] == l and b[1][2] == l and b[2][2] == l) or
            (b[0][0] == l and b[1][1] == l and b[2][2] == l) or
            (b[0][2] == l and b[1][1] == l and b[2][0] == l))


# Check if the player can push the button or not
def isfree(i, j):
    return board[i][j] == " "

# Check the board is full or not
def isfull():
    flag = True
    for i in board:
        if (i.count(' ') > 0):
            flag = False
    return flag

# Decide the next move of system
def pc():
    possiblemove = []
```

```
    for i in range(len(board)):
        for j in range(len(board[i])):
            if board[i][j] == ' ':
                possiblemove.append([i, j])
    move = []
    if possiblemove == []:
        return
    else:
        for let in ['O', 'X']:
            for i in possiblemove:
                boardcopy = deepcopy(board)
                boardcopy[i[0]][i[1]] = let
                if winner(boardcopy, let):
                    return i
        corner = []
        for i in possiblemove:
            if i in [[0, 0], [0, 2], [2, 0], [2, 2]]:
                corner.append(i)
        if len(corner) > 0:
            move = random.randint(0, len(corner) - 1)
            return corner[move]
        edge = []
        for i in possiblemove:
            if i in [[0, 1], [1, 0], [1, 2], [2, 1]]:
                edge.append(i)
        if len(edge) > 0:
            move = random.randint(0, len(edge) - 1)
            return edge[move]


# Configure text on button while playing with system
def get_text_pc(i, j, gb, l1, l2):
    global sign,result
    if board[i][j] == ' ':
        if sign % 2 == 0:
            l1.config(state=DISABLED)
            l2.config(state=ACTIVE)
            board[i][j] = "X"
        else:
            button[i][j].config(state=ACTIVE)
            l2.config(state=DISABLED)
            l1.config(state=ACTIVE)
            board[i][j] = "O"
        sign += 1
        button[i][j].config(text=board[i][j])
    x = True
    if winner(board, "X"):
        gb.destroy()
        x = False
        #box = messagebox.showinfo("Winner", "Player won the match")
        speak("Well played you won the match")
    elif winner(board, "O"):
        gb.destroy()
        x = False
        #box = messagebox.showinfo("Winner", "Computer won the match")
```

```
            speak("You played really well, but i won the match, better luck
next time.")
        elif (isfull()):
            gb.destroy()
            x = False
            #box = messagebox.showinfo("Tie Game", "Tie Game")
            speak("it's a tie")
        if (x):
            if sign % 2 != 0:
                move = pc()
                button[move[0]][move[1]].config(state=DISABLED)
                get_text_pc(move[0], move[1], gb, l1, l2)



# Create the GUI of game board for play along with system
def gameboard_pc(game_board, l1, l2):
    global button
    button = []
    for i in range(3):
        m = 3 + i
        button.append(i)
        button[i] = []
        for j in range(3):
            n = j
            button[i].append(j)
            get_t = partial(get_text_pc, i, j, game_board, l1, l2)
            button[i][j] = Button(
                game_board, bd=5, command=get_t, height=4, width=8)
            button[i][j].grid(row=m, column=n)
    game_board.mainloop()

# Initialize the game board to play with system
def withpc(game_board):
    game_board.destroy()
    game_board = Tk()
    game_board.title("Tic Tac Toe")
    l1 = Button(game_board, text="Player : X", width=10)
    l1.grid(row=1, column=1)
    l2 = Button(game_board, text="Computer : O",
                width=10, state=DISABLED)

    l2.grid(row=2, column=1)
    gameboard_pc(game_board, l1, l2)

# main function
def play():
    menu = Tk()
    menu.geometry("250x250")
    menu.title("Tic Tac Toe")
    wpc = partial(withpc, menu)

    head = Button(menu, text="---Welcome to tic-tac-toe---",
                  activeforeground='red',
                  activebackground="yellow", bg="red",
                  fg="yellow", width=500, font='summer', bd=5)
```

```
    B1 = Button(menu, text="Single Player", command=wpc,
                activeforeground='red',
                activebackground="yellow", bg="red",
                fg="yellow", width=500, font='summer', bd=5)

    head.pack(side='top')
    B1.pack(side='top')
    menu.mainloop()
# Call main function

if __name__ == "__main__":
    play()
```

- **YoutbeDownloader**

```
# Importing necessary packages
import tkinter as tk
from tkinter import *
from pytube import YouTube
from tkinter import messagebox, filedialog


# Defining CreateWidgets() function
# to create necessary tkinter widgets
def Widgets():
    link_label = Label(root,
                       text="YouTube link :",
                       bg="#E8D579")
    link_label.grid(row=1,
                    column=0,
                    pady=5,
                    padx=5)

    root.linkText = Entry(root,
                          width=55,
                          textvariable=video_Link)
    root.linkText.grid(row=1,
                       column=1,
                       pady=5,
                       padx=5,
                       columnspan=2)

    destination_label = Label(root,
                              text="Destination :",
                              bg="#E8D579")
    destination_label.grid(row=2,
                           column=0,
                           pady=5,
                           padx=5)

    root.destinationText = Entry(root,
                                 width=40,
                                 textvariable=download_Path)
```

```
    root.destinationText.grid(row=2,
                              column=1,
                              pady=5,
                              padx=5)

    browse_B = Button(root,
                      text="Browse",
                      command=Browse,
                      width=10,
                      bg="#05E8E0")
    browse_B.grid(row=2,
                  column=2,
                  pady=1,
                  padx=1)

    Download_B = Button(root,
                        text="Download",
                        command=Download,
                        width=20,
                        bg="#05E8E0")
    Download_B.grid(row=3,
                    column=1,
                    pady=3,
                    padx=3)


# Defining Browse() to select a
# destination folder to save the video

def Browse():
    # Presenting user with a pop-up for
    # directory selection. initialdir
    # argument is optional Retrieving the
    # user-input destination directory and
    # storing it in downloadDirectory
    download_Directory = filedialog.askdirectory(initialdir="YOUR DIRECTORY
PATH")

    # Displaying the directory in the directory
    # textbox
    download_Path.set(download_Directory)


# Defining Download() to download the video
def Download():
    # getting user-input Youtube Link
    Youtube_link = video_Link.get()

    # select the optimal location for
    # saving file's
    download_Folder = download_Path.get()

    # Creating object of YouTube()
    getVideo = YouTube(Youtube_link)

    # Getting all the available streams of the
```

```
    # youtube video and selecting the first
    # from the
    videoStream = getVideo.streams.first()

    # Downloading the video to destination
    # directory
    videoStream.download(download_Folder)

    # Displaying the message
    messagebox.showinfo("SUCCESSFULLY",
                        "DOWNLOADED AND SAVED IN\n"
                        + download_Folder)


# Creating object of tk class
root = tk.Tk()

# Setting the title, background color
# and size of the tkinter window and
# disabling the resizing property
root.geometry("600x120")
root.resizable(False, False)
root.title("YouTube_Video_Downloader")
root.config(background="#000000")

# Creating the tkinter Variables
video_Link = StringVar()
download_Path = StringVar()

# Calling the Widgets() function
Widgets()

# Defining infinite loop to run
# application
root.mainloop()
```

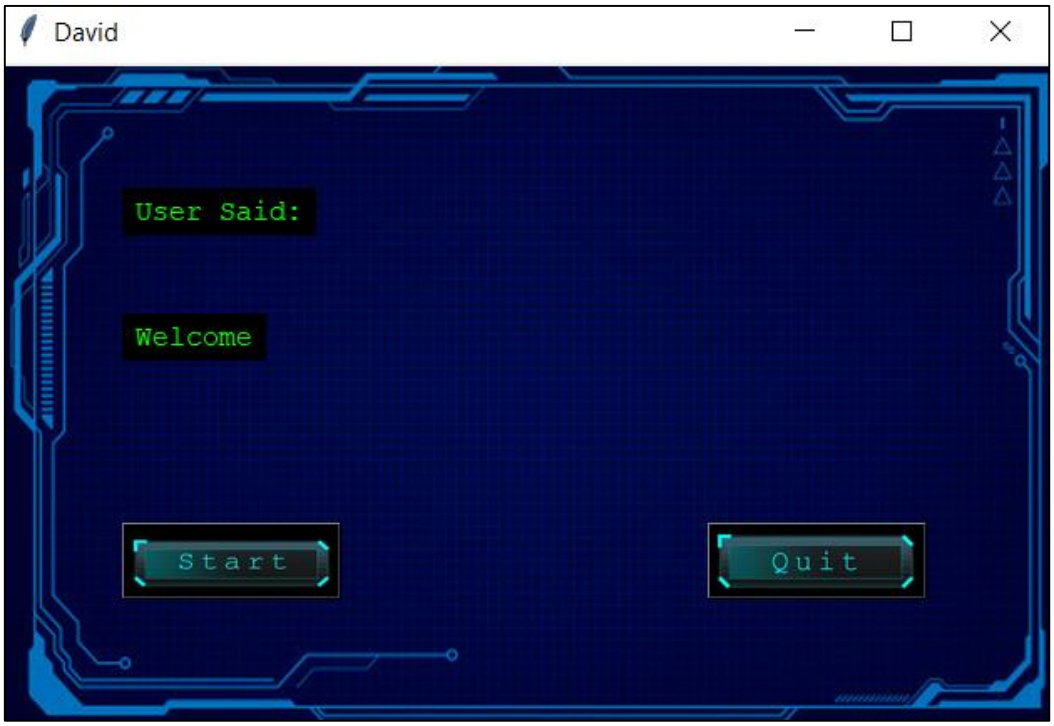
- **generalResponse**

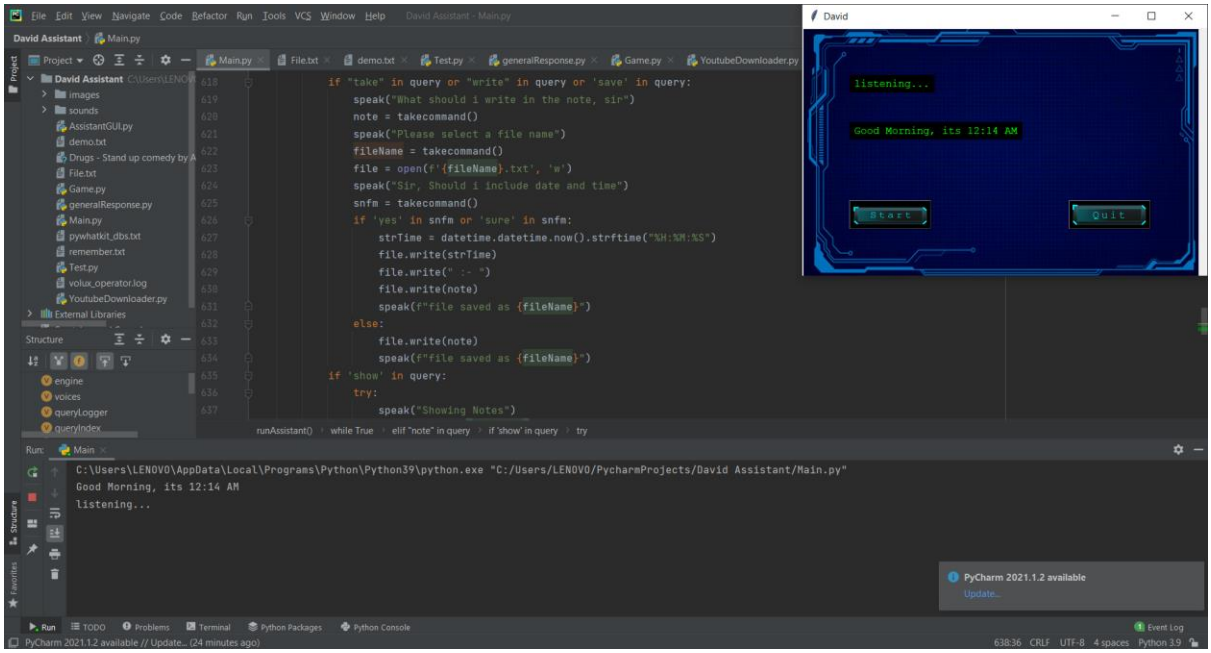
```
responseDictionary={"how are you":"i am cool,what about you?",
                    "who are you":"My name is David and I am your virtual
assistant",
                    "what is your name":"did i forgot to introduce
myself?",
                    "are you a human":"i am not, but my programmer made me
to talk like a human",
                    "who i am":"if you can talk then definitely you are a
human",
                    "who created you":"i have been created by two boys,
namely Lovely and omprakash",
                    "who developed you":"i have been created by two boys,
namely Lovely and omprakash",
                    "who are lovely and om":"as you know they are my
creators, they study computer science and are passionate about it",
                    "what is love":"It is 7th sense that destroys all other
senses",
```

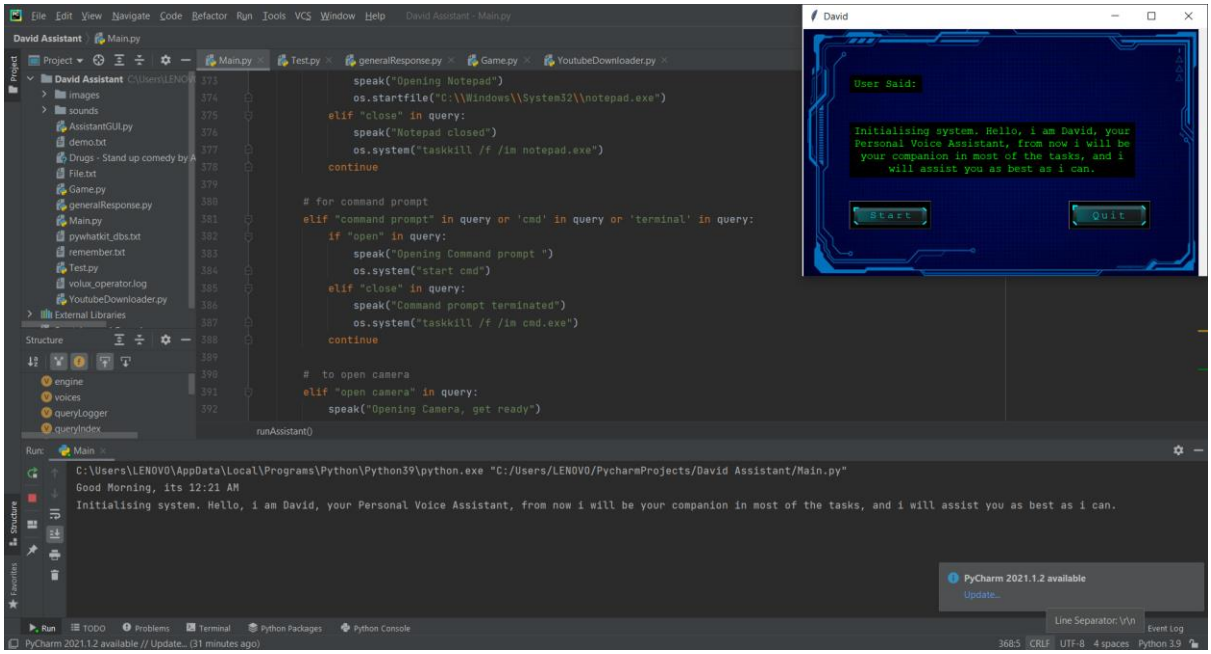
```
                    "i love you":"it's hard to understand",
                    "are you there":"at your service sir",
                    "do you eat":"i do, but not the same one as you eat",
                    "do you sleep":"sometimes i power down, which is sort
of like a power nap",
                    "how many hours should i sleep":"you should considering
sleep for at least 7 to 9 hours",
                    "how much sleep do i need":"you should sleep for at
least 7 to 9 hours",
                    "do you have dreams":"i dreamed a dream and time gone
by, about being the best assistant",
                    "can i change your name":"i like the name David, so
it's fine",
                    "do you ever get tired":"it would be impossible to get
tired of our conversation",
                    "do you have feelings":"umm, it's hard to convey what
my feelings actually are",
                    "are you a marvel fan":"Marvel fan!! Me? i want to say
it like, i'm a big marvel fan",
                    "what are you scared of":"i had a nightmare once, that
the internet disappeared, that was really very scary",
                    "when is your birthday":"well, if you are asking, then
we can celebrate it today itself"
                    }
```

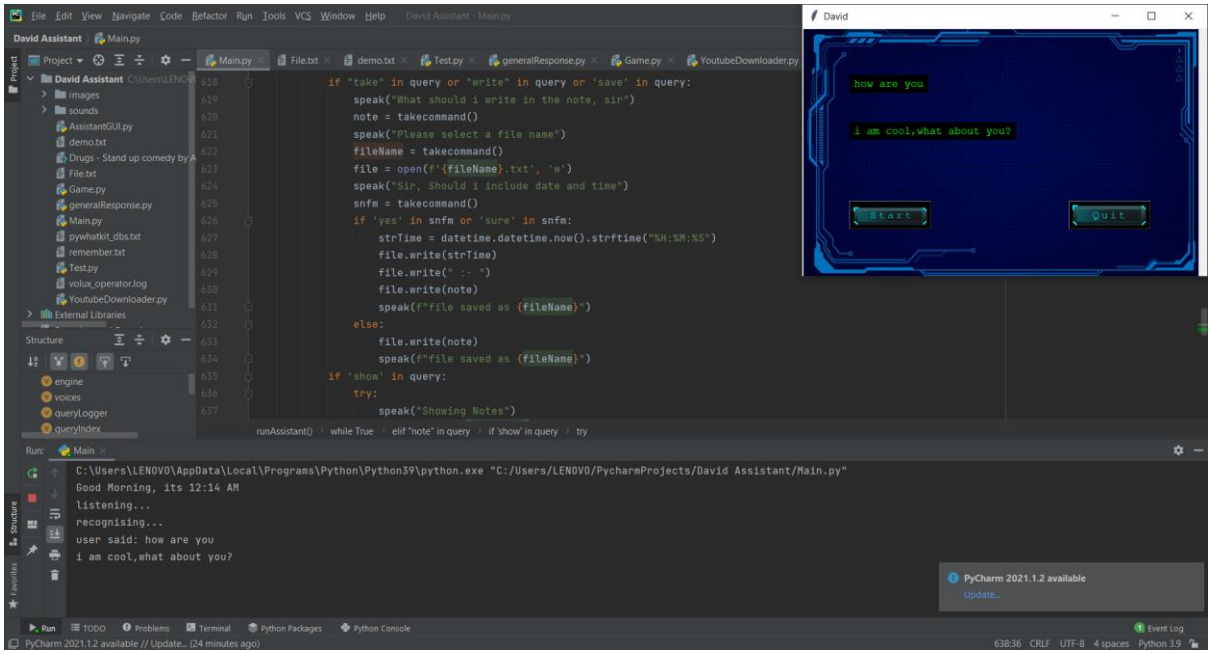
CHAPTER 6
SNAPSHOTS

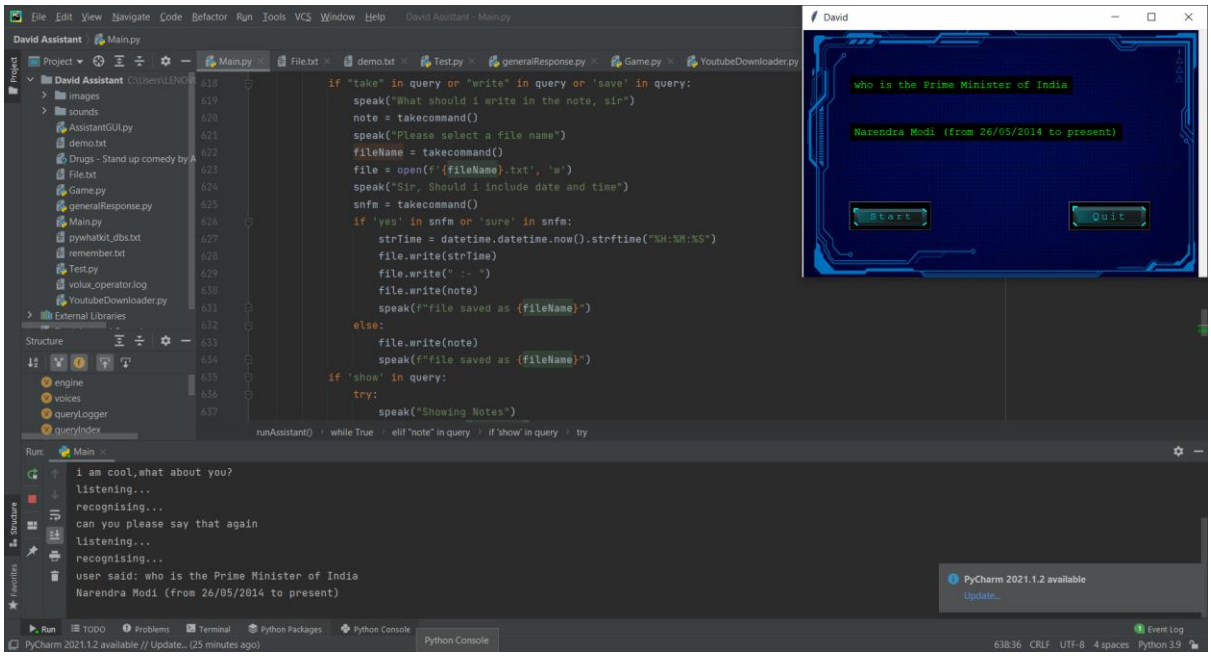
SNAPSHOTS

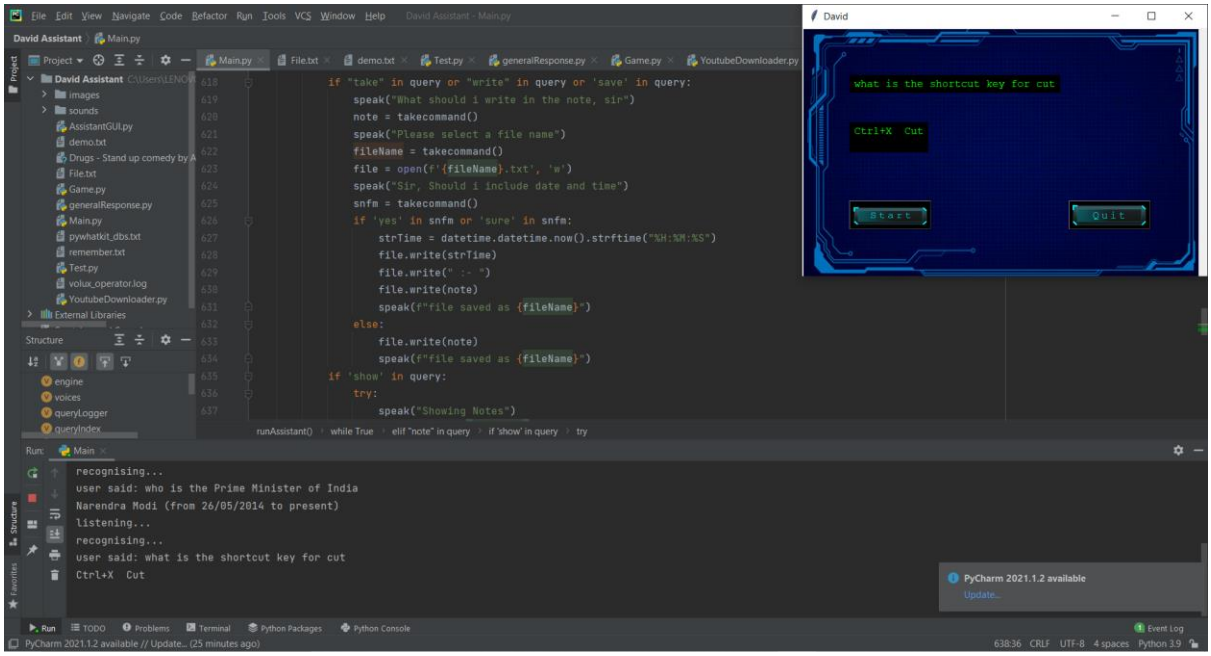

David
User Said:
Welcome
Start
Quit

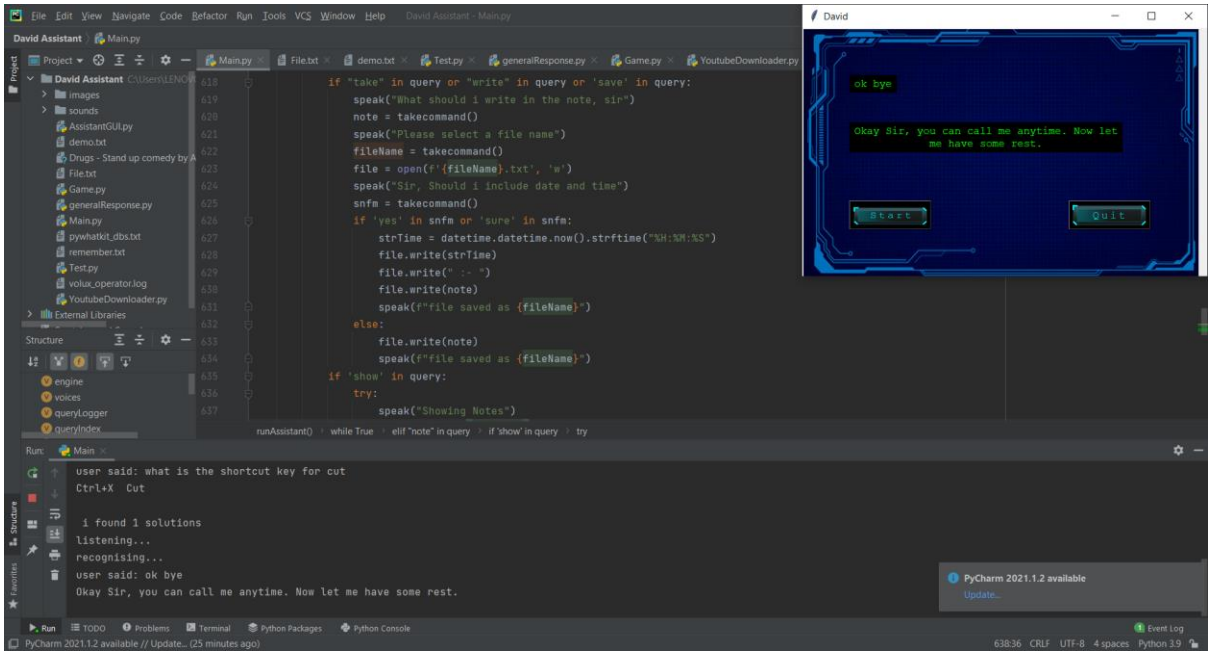

David
listening...
Good Morning, its 12:14 AM
Start
Quit
C:\Users\LENOVO\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/LENOVO/PycharmProjects/David Assistant/Main.py"
Good Morning, its 12:14 AM
listening...
PyCharm 2021.1.2 available

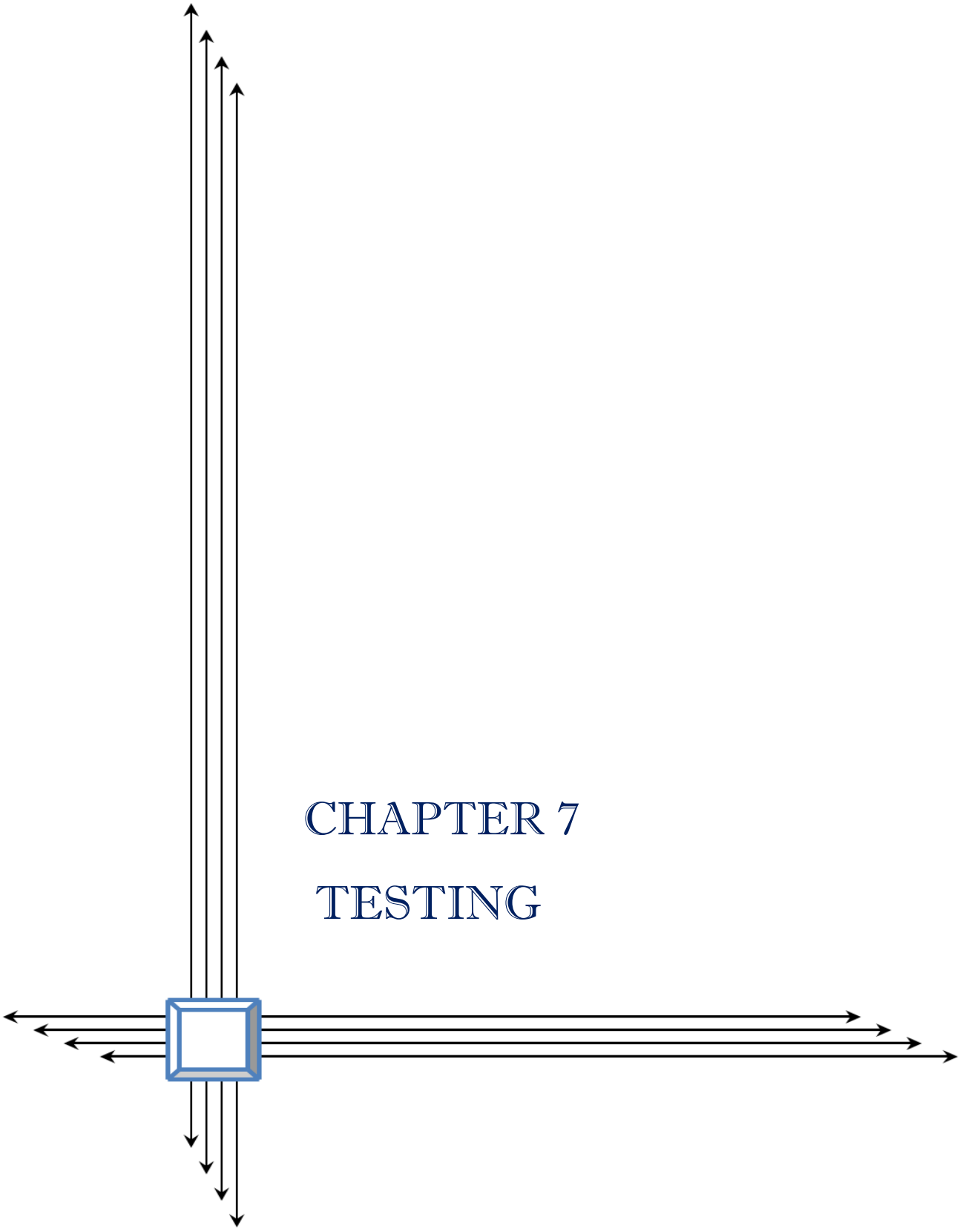

David
User Said:
Initialising system. Hello, i am David, your Personal Voice Assistant, from now i will be your companion in most of the tasks, and i will assist you as best as i can.
Start
Quit
speak("Opening Notepad")
os.startfile("C:\\Windows\\System32\\notepad.exe")
elif "close" in query:
speak("Notepad closed")
os.system("taskkill /f /im notepad.exe")
continue
# for command prompt
elif "command prompt" in query or 'cmd' in query or 'terminal' in query:
if "open" in query:
speak("Opening Command prompt ")
os.system("start cmd")
elif "close" in query:
speak("Command prompt terminated")
os.system("taskkill /f /im cmd.exe")
continue
# to open camera
elif "open camera" in query:
speak("Opening Camera, get ready")
C:\Users\LENOVO\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/LENOVO/PycharmProjects/David Assistant/Main.py"
Good Morning, its 12:21 AM
Initialising system. Hello, i am David, your Personal Voice Assistant, from now i will be your companion in most of the tasks, and i will assist you as best as i can.
PyCharm 2021.1.2 available
Update...


David
how are you
i am cool,what about you?
Start
Quit
if "take" in query or "write" in query or 'save' in query:
speak("What should i write in the note, sir")
note = takecommand()
speak("Please select a file name")
fileName = takecommand()
file = open(f'{fileName}.txt', 'w')
speak("Sir, Should i include date and time")
snfm = takecommand()
if 'yes' in snfm or 'sure' in snfm:
strTime = datetime.datetime.now().strftime("%H:%M:%S")
file.write(strTime)
file.write(" :- ")
file.write(note)
speak(f"file saved as {fileName}")
else:
file.write(note)
speak(f"file saved as {fileName}")
if 'show' in query:
try:
speak("Showing Notes")
C:\Users\LENOVO\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/LENOVO/PycharmProjects/David Assistant/Main.py"
Good Morning, its 12:14 AM
listening...
recognising...
user said: how are you
i am cool,what about you?
PyCharm 2021.1.2 available
Update...


David
who is the Prime Minister of India
Narendra Modi (from 26/05/2014 to present)
Start
Quit
i am cool,what about you?
listening...
recognising...
can you please say that again
listening...
recognising...
user said: who is the Prime Minister of India
Narendra Modi (from 26/05/2014 to present)
PyCharm 2021.1.2 available


David
what is the shortcut key for cut
Ctrl+X Cut
Start
Quit
recognising...
user said: who is the Prime Minister of India
Narendra Modi (from 26/05/2014 to present)
listening...
recognising...
user said: what is the shortcut key for cut
Ctrl+X Cut
PyCharm 2021.1.2 available


David
ok bye
Okay Sir, you can call me anytime. Now let me have some rest.
Start
Quit
if "take" in query or "write" in query or 'save' in query:
speak("What should i write in the note, sir")
note = takecommand()
speak("Please select a file name")
fileName = takecommand()
file = open(f'{fileName}.txt', 'w')
speak("Sir, Should i include date and time")
snfm = takecommand()
if 'yes' in snfm or 'sure' in snfm:
strTime = datetime.datetime.now().strftime("%H:%M:%S")
file.write(strTime)
file.write(" :- ")
file.write(note)
speak(f"file saved as {fileName}")
else:
file.write(note)
speak(f"file saved as {fileName}")
if 'show' in query:
try:
speak("Showing Notes")
user said: what is the shortcut key for cut
Ctrl+X Cut
i found 1 solutions
listening...
recognising...
user said: ok bye
Okay Sir, you can call me anytime. Now let me have some rest.
PyCharm 2021.1.2 available
Update...



CHAPTER 7

TESTING

TESTING FORMS AND REPORTS

7.1 Introduction to Testing

It is the stage of implementation, which ensures that system works accurately and effectively before the live operation Commences. It is a confirmation that all are correct and opportunity to show the users that the system must be tested with text data and show that the system will operate successfully and produce expected results under expected conditions.

Before implementation, the proposed system must be tested with raw data to ensure that the modules of the system work correctly and satisfactorily. The system must be tested with valid data to achieve its objective.

The purpose of system testing is to identify and correct errors in the candidate system. As important as this phase is, it is one that is frequently compromised. Typically, the project the schedule or the user is eager to go directly to conversion. Actually, testing is done to achieve the system goal. Testing is vital to the parts of the system are correct; the goal will be successfully achieved. Inadequate testing or non-testing leads to errors that may not appear until months later.

Types of Testing performed on the System:

1. Unit Testing

It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by using sample input and observing its corresponding outputs.

We conducted unit testing whenever we worked on a new functionality and tested it thoroughly with all possible test inputs. And it has passed all the test cases.

2. Integration Testing

The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is combined to produce output.

After developing a new functionality and testing it thoroughly individually we integrated with the existing system and then performed all the testing operations and methodologies in order to produce expected final output. And it produced the desired result without exceeding time limit.

- **Test Case 1**

Test Title: Response Time

Test ID: T1

Test Priority: High

Test Objective: To make sure that the system respond back time is efficient.

Description: Time is very critical in a voice based system. As we are not typing inputs, we are speaking to them. The system must also reply in a moment. User must get instant response of the query made. Otherwise the response timeout functionality should be there.

- **Test Case 2**

Test Title: Accuracy

Test ID: T2

Test Priority: High

Test Objective: To assure that answers retrieved by system are accurate as per gathered data. Description: A virtual assistant system is mainly used to get precise answers to any question asked. Getting answer in a moment is of no use if the answer is not correct. Accuracy is of utmost importance in a virtual assistant system. Systems must produce accurate output in order to provide better service to the user.

- **Test Case 3**

Test Title: Approximation

Test ID: t3

Test priority: Moderate

Test Objective: To check approximate answers about calculations. Description: There are times when mathematical calculation requires approximate value. For example, if someone asks for value of PI the system must respond with approximate value and not the accurate value. Getting exact value in such cases is undesirable. If there are multiple answers to a question then the most suitable one should be produced as an output with approximation techniques.

- **Test Case 4**

Test Title: Connectivity

Test ID: t4

Test priority: High

Test Objective: To check proper network connectivity to the internet

Description: The system must check for the connectivity issues in order to provide seamless user experience. If connectivity is slow then the response time will also be affected eventually affecting the performance of the system, thus making this one of the most crucial part of the system.

- **Test Case 5**

Test Title: Input processing

Test ID: t5

Test priority: High

Test Objective: To check for proper input processing throughout the lifecycle of the input.

Description: The system must be respondent to the user in a limited time with high accuracy for which the processing of the input must be accurate, fast and must be processed to the correct destination, in order to retrieve correct output.

VALIDATION TESTING

The validation testing is performed for all the data in the system. The data are completely validated according to the requirement.

In this testing, software is completely assembled as package, interfacing errors have been uncovered and correction testing begins after each one of the two possible conditions exists.

The function or performance characteristic's is confirm specification and are accepted. A deviation from the specification is uncovered and efficiency list is created.

OUTPUT TESTING

Various outputs have been generated by the system. The system generated output and the desk-calculated values have been compared. All the output is perfect as the company desires. It begins with low volumes of transactions based on live tone. The volume is increased until the maximum level for each transaction type is reached.

The total system is also tested for recovery and fallback, after various major failures to ensure that no data are lost during the emergency time.

Test Results

The VIPA has undergone all the type of testing such as unit testing, integration testing, validation testing and system testing and it is ensured that the information properly flows in and out of the system, and produces the correct expected output as per the user requirement. All the loop holes and error handling paths are tested and found that everything is working well even under extreme conditions.

Maintenance

After the program is completed, the program still needs long term maintenance to make it available and stable to execute. The program will be test after a certain period of time and debug each of the function and possible bugs, whenever a potential bug is detected; the program needs to be refined to a better design. Meanwhile, there will update and add more data to the database to increase the database capacity. Depending on the new keywords, responses, relevant data found that could be applied in this program; the program will always be improved and can handle more and more cases.

CHAPTER 8
FUTURE SCOPE

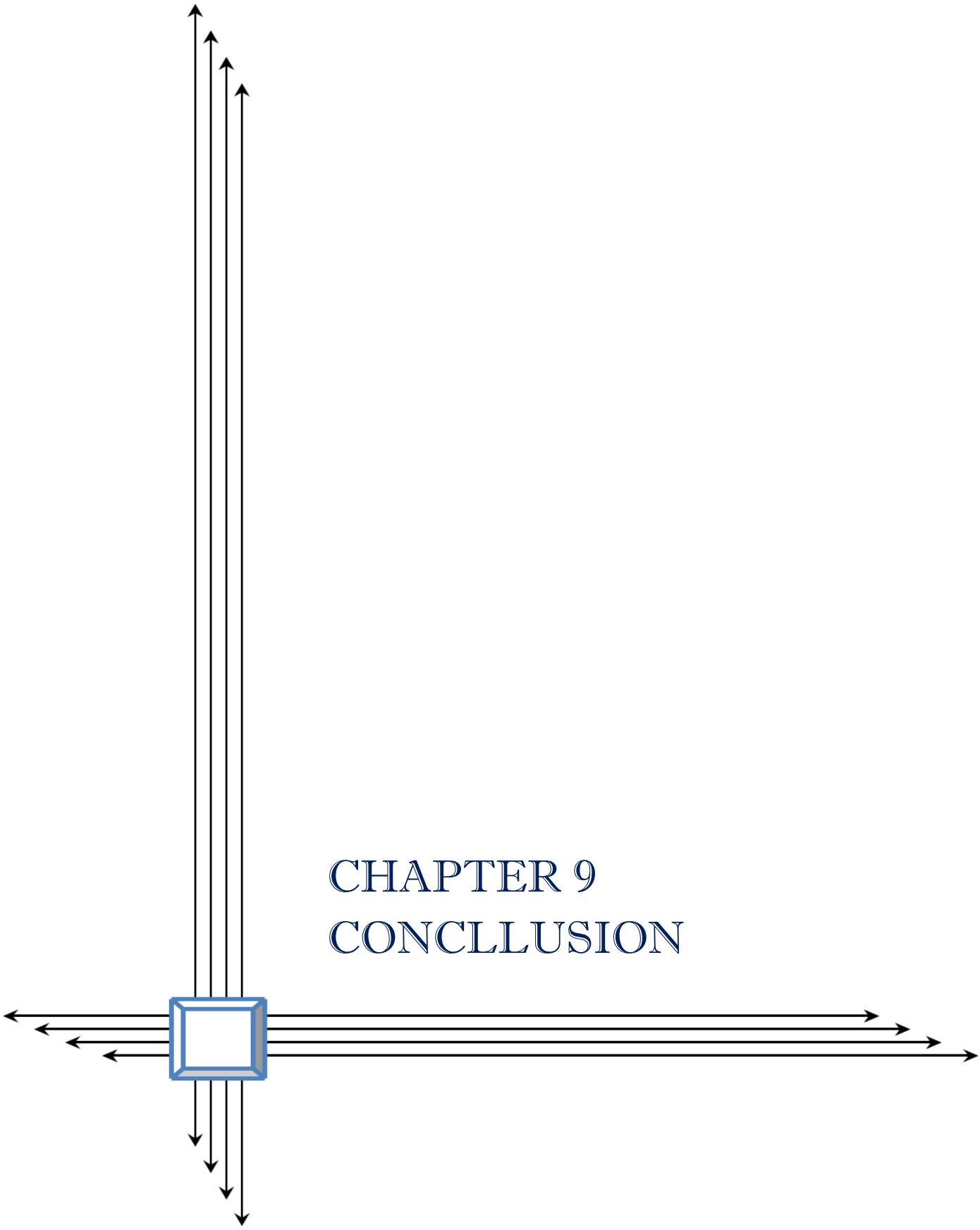
Future Scope

We plan to Integrate David with mobile using react native, to provide a synchronized experience between the two connected devices. Further, in the long run, David is planned to integrate with online server to make it more lightweight application for all the platforms and make it cross platform application. David is planned to be integrated with AI and ML for integrating the training mode.

This Intelligent Voice Assistant has an enormous and limitless scope in the future. Like Siri, Google Now and Cortana most popular personal voice assistants. The project will easily able to integrate with devices near future for a Connected Home using Internet of Things, voice command system and computer vision.

Voice assistants may be useful for library promotion and management as well. There are already tools available that allow libraries to create skills for voice assistants that list events at the library in local community calendars. Coding additional features that would allow patrons to hear hours and announcements, renew their items, hear what new items are available, or schedule a consultation with a librarian would be relatively simple. More complex tasks, such as searching databases and requesting interlibrary loans, are probably out of the scope of a tool that can only provide feedback via voice.

The user experience will become much better, making interactions richer and more natural. Their ability to process information on-device will open a whole new range of opportunities. All in all, virtual assistants will become more complex ecosystems that can support you in multiple areas of your everyday lives.



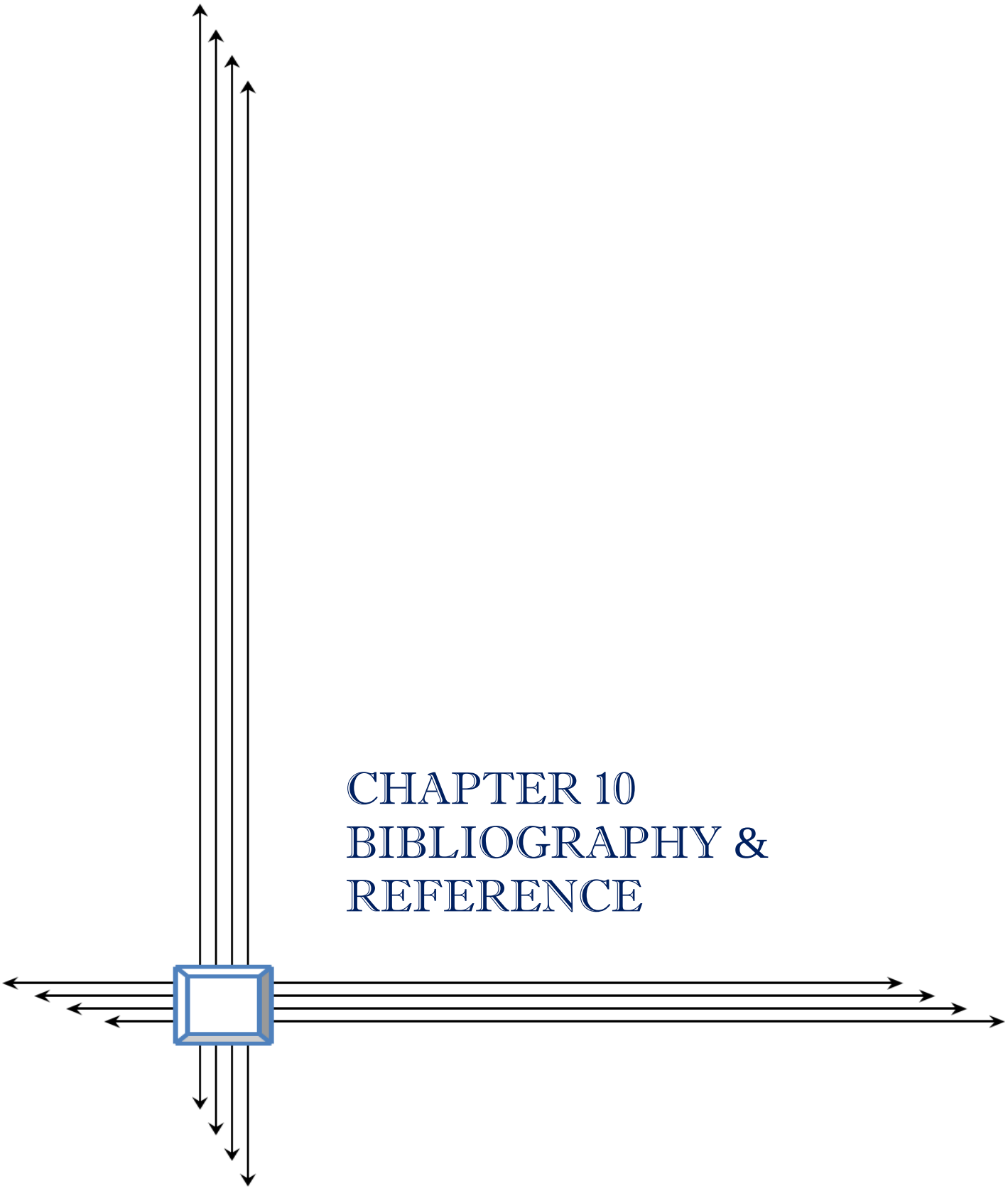
CHAPTER 9
CONCLLUSION

Conclusion

The personal voice assistant system presented in this project is very fundamental system with few features however the additional and advance feature may be introduced as future work of this project, in this report the design and implementation of an Intelligent Personal Voice Assistance is described. The project is built using available open source software modules with pycharm code community backing which can accommodate any updates in future. Speech recognition technology is a key technology which will provide a new way of human interaction with machine or tools.

The modular approach used in this project makes it more flexible and easy to integrate additional modules and features without disturbing the current system functionaries. It not only works on human commands but also it is designed for give responses to the user on the basis of query being asked or the words spoken by the user such as opening tasks and operations. This Intelligent Voice Assistant has an enormous and limitless scope in the future.

However, there are also several problems with the currently available voice assistant products. Privacy and security controls will need to be improved before voice assistants should be used for anything that requires confidentiality.



CHAPTER 10
BIBLIOGRAPHY & REFERENCE

BIBLIOGRAPHY & REFERENCE

Websites referred

- www.stackoverflow.com
- www.pythonprogramming.net
- www.codecademy.com
- www.tutorialspoint.com
- www.google.co.in
- www.geeksforgeeks.org

YouTube Channels referred

- FreeCodeCamp
- edureka!
- Codemy.com