**UNIVERSITY INSTITUTE OF COMPUTING**

**DIVISION- MCA**

A Project Report

On

CHATBOT HELPER.

Submitted in partial fulfillment of the requirements for the award of the degree of

**(MCA)**

**In**

**Session: 2021**

**By**

**Parth Kushwaha**

**(UID-20MCA1210)**

**Supervised By**

**Bachandeep Singh Bhathal**

DESIGNATION



**University Institute of Computing**

**CHANDIGARH UNIVERSITY, GHARUAN, MOHALI, PUNJAB,140413**

**Acknowledgement**

I would like to express my special thanks of gratitude to my teacher Mrs. Rinku Sharma who gave me the special opportunity to do this project of Chatbot helper.

Who has always helped in completing my project. I came to know about many new and wonderful things while doing this project. Also, it would not have been possible without the kind support and help of many individuals and the institution **University of Computing (UIC)**, **Chandigarh University, Mohali, Punjab**.

It’s my radiant point of view to place on record our best regards and deepest sense of gratitude to all members of UIC, Chandigarh University, Mohali, Punjab for their careful and invaluable guidance which were extremely valuable for our project.

I would also like to express our heartfelt gratitude towards respected **Rydhm Beri, UIC, Chandigarh University, Mohali, Punjab** for providing us this opportunity and necessary information regarding the project work.

**Candidates Declaration**

I hereby corroborate that the work which is being presented in the project titled “CHATBOT HELPER” in partial fulfilment of the requirements for the award of the Master of Computer Applications and submitted in the University of Computing (UIC), Chandigarh University, Mohali, Punjab, is an authentic record of my own work completed during a period from August 2021- December 2021 under the supervision of Mrs. Rinku Sharma, Assistant Professor, UIC, Chandigarh University, Mohali, Punjab.

The matter presented in this project has not been submitted by me for the reward of any other degree of this or any other Institute/University.

**Name of Candidate:**

Parth Kushwaha

Uid:- 20MCA1210

**Training Letter**

**Certificate by Supervisor**

**Abstract**

The main purpose of this project is to create a chatbot that can help a user with its day-to-day tasks in every way. The simple use of chatbot can be that it can be used in everyday life. A person that doesn’t know much about computer or how to use it can simply give commands through his/her voice or they can use the keyboard to type in their requests. They can used for many purposes like to play a song or to use internet or to search a keyword on Wikipedia. Mainly it is a combination of both a chatbot and a voice assistant. Though there is major difference between both of them. The chatbot is mainly used to type and search the keywords or sentences and then give a fixed result. Because of that there a few limitations it can only show the result that are already stored in the system. But after connecting it with voice assistant and internet it improves drastically and can do many other things. This paper will project light on these things.

**Preface**

The goal is to improve the old chat bots and give them a new look according to the modern world. Describing in simple words it can basically change the perspective of the chatbots in the modern world. The simple use of chatbot can be that it can be used in everyday life. A person that doesn’t know much about computer or how to use it can simply give commands through his/her voice or they can use the keyboard to type in their requests. They can used for many purposes like to play a song or to use internet or to search a keyword on Wikipedia. Mainly it is a combination of both a chatbot and a voice assistant. Though there is major difference between both of them. The chatbot is mainly used to type and search the keywords or sentences and then give a fixed result. Because of that there a few limitations it can only show the result that are already stored in the system. But after connecting it with voice assistant and internet it improves drastically and can do many other things. It looks way more futuristic and is way more useful than a simple chatbot.

**Table of Content**

|  |  |
| --- | --- |
| **Topic** | **Page No.** |
| Introduction to Project | 9 |
| OBJECTIVE | 10 |
| SCOPE OF THE PROJECT | 11 |
| FEATURES OF THE PROJECT | 12 |
| APPLICABLITY | 12 |
| Organization of Report | 13 |
| SYSTEM ANALYSIS | 14 |
| FEASIBILITY STUDY | 15 |
| PROJECT PLANNING | 16 |
| PROJECT SCHEDULING | 17 |
| Software Requirement | 18 |
| Software Engineering Paradigm | 19 |
| DATA DLOW DIAGRAM | 20-22 |
| System Design | 23 |
| Coding | 24-67 |
| Standardization of the coding | 68-70 |
| Testing | 71-74 |
| Documentation | 75 |
| Conclusion | 76 |
| Future Scope | 77 |
| REFRENCES | 78 |

**INTRODUCTION**

A chatbot is a smart piece of software that can communicate and perform human-like actions. Chatbots are widely used in customer interaction, marketing on social networking sites, and instant messaging to the client. There are two basic types of discussion models depending on how they are constructed; Return based on Modified and Manufactured models. The first chatbot was created by Joseph wiesenbaum in the year 1966 and was given the name ELIZA. Today there are many types of virtual assistants available in the industry that helps the user in every aspect whether it be asking for the basic search or playing music or doing calculations in calculator.The use of this project is to create a chatbot that can help a user in his/her daily tasks. Theat can handle all the work without any problem. It basically does all the work for the user just by receiving the commands either through voice command or through keyboard. It is a bit different from other chatbots that have only one function through which you can give commands (either through voice or typing). What makes it really different from others is that it can perform tasks in the system as well as from the internet.

**OBJECTIVE**

The main objective of this project is to help the costumers with their problems that they face during surfing a website. When a customer wants to get any information about any product or wants to know about the offers that the website is giving to their customers. The objective of this project is to make a chatbot using python and flask using concepts of machine learning. It learns from its experiences and adapt according to the situations presented in front of it.

**Purpose of the Project**

The goal is to provide a particular type of chatbot that can help the user in many different ways. By different it means to deal with the basic things that a normal user does in day-to-day life.

It provides assistance in both type of commands:

* Through voice commands
* By using keyboard

**SCOPE OF THE PROJECT**

Just before it becomes a common customer support to replace people, negotiations have emerged with a wave of artificial intelligence.

With each passing day, they become more intelligent and involved. Undoubtedly the face of negotiations will change drastically in 2020 but what could they be.

**Chatbots replaces apps**

* The world has seen about 6.5 million applications being developed. It has been observed that 23% of users uninstall apps after a few weeks of use.
* Businesses lose many businesses in this way. Chatbots came together as a potential way to engage with the audience at the time. The messaging app brings an easy-to-download chatbot and users engage with campaigns instantly.

**Emotional performance**

* Improved conversations are just a good line of being able to deliver a personal-level conversation. They have already won over the natural language correction zone and the next in scope is NLU and NLG.
* NLG enters the picture while the chatbot tries to generate an answer to the question. It depends on the day-to-day operation and the next dimension is the response of many languages.

**Voice bots**

* Chabot's has received a lot of criticism from the developers' category who believe they should have a user interface. They prefer the voice to the tap.

**FEATURES OF THE PROJECT**

* Today chatbots are improving efficiency.
* Chatbot reduces the dependence on humans.
* They can smartly answer the questions in real-time.
* They are mainly used for interacting with costumers.
* In this project we have tried to turn the chatbot helper in an automatic system so that it can learn and provide help accordingly.
* It will learn from its past interactions and then improve itself.

**APPLICABLITY**

The use of this project is to create a chatbot that can help a user in his/her daily tasks. Theat can handle all the work without any problem. It basically does all the work for the user just by receiving the commands either through voice command or through keyboard. It is a bit different from other chatbots that have only one function through which you can give commands (either through voice or typing). What makes it really different from others is that it can perform tasks in the system as well as from the internet.

For example, doing calculation, setting timer, playing games(rock, paper scissor.) as well as on internet it can make a search for the user(covid tracks, weather, play videos etc.)

**Organization of Report**

The material presented at the workshop is organized into seven chapters. After this introductory chapter.

* Chapter 2 describes the analysis of the system their identification, if the model Is sustainable or not it’s feasibility. Also deals with the planning of the project and its scheduling. There is SRS (software requirement specification) in this chapter. Most of the structured programming part is the issue of this chapter. Like DFD model, class diagram, etc.
* Chapter 3 summarizes the module summary, the areas of constraints in the project, OOP design, test different types of test cases. Like unit test and system test case.
* Chapter 4 provides the hands-on coding example and the part of the coding that was in the project. Comments and description of what exactly is happening in the project, what coding standard we are using, how efficient is our code. How did we handled the error faced during the program runtime and how we handled exception handling
* Chapter 5 presents the testing phase of the report and gets into the nitty gritty of the different testing scenarios. What plans were used and how it was processed and handled by the user. Debugging if we used any. Database structure and the structure of the security measure taken up by the developer.
* Chapter 6 is one of the final stages of the report and contains stuff such as the cost sustained during the development of the project. The pre-requisite required for the installation of the project and comparative analysis between our project and the other competition on the market right now.
* Chapter 7 contains the conclusion and the future scope (if any available) for the project.

**SYSTEM ANALYSIS**

**Identification' of Need**

The main use of this project is to help the costumers with their problems that they face during surfing a website. When a customer wants to get any information about any product or wants to know about the offers that the website is giving to their customers. The objective of this project is to make a chatbot using python and flask using concepts of machine learning. It learns from its experiences and adapt according to the situations presented in front of it.

It is a bit different from other chatbots that have only one function through which you can give commands (either through voice or typing). What makes it really different from others is that it can perform tasks in the system as well as from the internet.

**FEASIBILITY STUDY**

This project is mainly based on machine learning, which requires a system with basic CPU to work through the training process.

* **Technical Feasibility**

The minimum specifications are given below:

* CPU – Quad core with 8 threads to not bottleneck the GPU/ working with AMD Ryzen 5 35000h
* GPU—Nvidia GTX 940MX / working with GTX 1650
* Storage – 500 GB SATA Hard disk / working with 500 GB nvme SSD
* Ram – 4GB / working with 8GB

* **End User Feasibility**

This project is relatively simple for anyone with a very basic knowledge to operate and work upon. Steps and measures are being taken to make it more user Feasible but for now it is relatively simple, in future we hope for extreme end user feasibility.

**PROJECT PLANNING**

The basic work for a chatbot is to provide help and assistance to a user on the computer. The main planning for this project was to how to design it so that it can easily be understandable for the user the one who will use it. It is simple task to say but difficult to execute. For the execution of this creation languages and libraries were used. The main was python and OpenCV. For the Gui Tkinter was used. This project was mainly made to describe how Python and opcv can be used to make a chatbot that is much better than those are already available in the market.

There were a few things that were new for me but with the help of fellow classmates and teacher it was made possible. The main goal was to create a chatbot that can be used by everyone whether a user is new to Computer or not. It is made with full effort to keep it simple and easy to use for everyone.

**PROJECT SCHEDULING**

**Week 1**

* Project understanding
* Getting domain knowledge
* Understanding the dataset (images and csv file)

**Week 2**

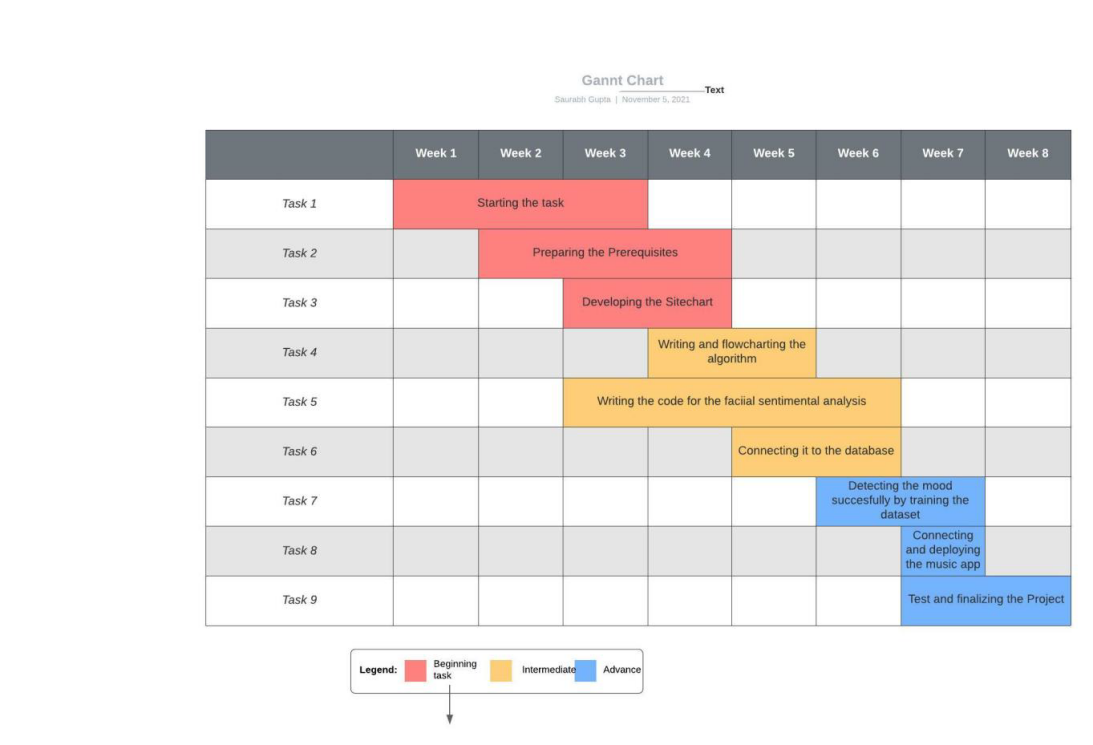
* Exploratory data analysis
* Data augmentation

**Week 3**

* Building of machine learning model (CNN Layers)
* Training the dataset with various parameters (activation functions and base models)

**Week 4**

* Comparative analysis for accuracy and loss based on different batch sizes.
* Obtaining the final model and csv file with probability of metastatic tissue present for the testing image dataset.



**Software Requirement:**

* **OpenCV** (Open-Source Computer Vision Library) is an open-source computer vision and ML software library in Python programming language. This was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easier for businesses to utilize and change the code. The library consists of more than 2500 optimized algorithms, which comprises an extensive set of both classic and state of the art computer vision and machine learning algorithms. These algorithms can be used to detection and recognizing faces, identifying objects, classifying human actions in videos, tracking camera movements, tracking moving objects, extracting 3D models of objects, producing 3D point clouds from stereo cameras, stitching images together to produce a high resolution image of an entire site, finding similar images from an image database, removing red eyes from images taken using flash, following eye movements, recognizing scenery and establishing markers to overlay it with augmented reality, etc.
* **TKinter** is a standard GUI based library for python. When combined with python, it provides a fast and easy wat to create GUI based applications. TKinter provides a powerful OOP’s interface to the TK GUI toolkit.it is not a thin wrapper but it also adds a good amount of its own logic to make a experience pythonic.
* **Pyttsx3** is a text to speech library available in python. It basically converts text in speeches form. It is also works in offline mode and is very much compatible with both the versions of python (2&3). It invokes pyttsx3.init() to get a reference. It is a very easy tool to convert text to speech.
* **Python** is a high level language which is used for general purposes as well as high end language. It is dynamically typed and garbage collected. It supports OOP’s concept and is considered a dynamic that means it can work for almost everything.

**Hardware**

* Windows 10/Linux OS.
* CPU in comparison to Ryzen 3 3rd gen or Intel I3 5 th gen.
* Any GPU that is compatible with OpenGL 3.2. (Integrated graphics cards Intel
* HD 4000 or above).
* 4 GB RAM, 10 GB HDD Free Space.

**Software**

* An Operating System
* A Web Browser Higher than 2011 Models
* Python 3.9

**Software Engineering Paradigm applied**

SDLC is an abbreviation that stands for software development life cycle. It is referred to as the

software development process in some circles. All of the tasks involved in the development and

maintenance of software. It is a process that includes the steps of planning, developing, testing, and implementing an information system. These steps include requirements gathering, validation, training, and ownership.

The stages of the SDLC model are as follows:

Following is a brief review of the SDLC steps that you should be familiar with.

* Stage 1: Requirements collecting and analysis —

The feasibility report is favorable to the project, and the following step will begin with the

gathering of requirements from the end-users. Engineers connect with clients and end-users to

learn about their ideas and which additions they would want to see included in the product.

* Stage 2: Software design — This is the final stage.

It is the process of transforming user needs into a form that can be implemented. It is beneficial

to programmers when coding software. In software, there is a necessity for more exact and

detailed requirements to be specified. The output of the procedure can be utilized directly in

the implementation of a program in a programming language without further processing. There

are three tiers of design, which are as follows.Architectural design - It is the most abstract version of the system at the highest level of abstraction. A software system is made up of various components that communicate with one another.

High-level design — This type of design is concerned with how the system, including all of its

components, can be implemented in the form of modules.

* Stage 3: - development of the product.

You will learn about the product development process at this phase of the SDLC. It is one of the

most important phases of the SDLC, and it is also referred to as the Implementation phase.

* Stage 4: Testing and integration of the product are carried out during this stage. We will

integrate the modules and test the whole product using a variety of testing approaches. We will

also perform maintenance on the product in order to prepare it for any future updates and the

release of new features.

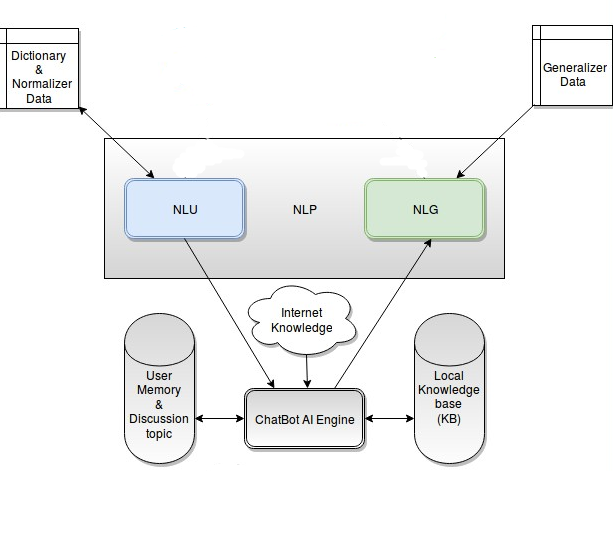
* Stage-5: Product Deployment and Maintenance – During this phase, we will perform actual

product deployment, or you can say the final product deployment, as well as product

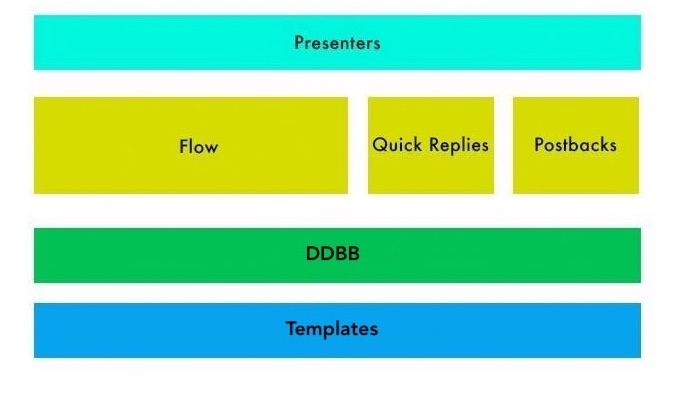
maintenance in preparation for any future updates and release of new features.

**Data models (DFD)**

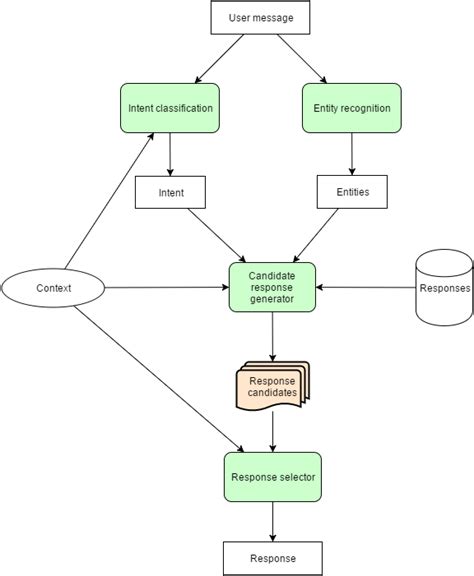
**Chatbot-retrieval-based-model**



**Basic architecture of a chatbot**



**DATA DLOW DIAGRAM**



**System Design**

**Modularization details**

In software engineering, modularization refers to the process of breaking up a software system

into a number of discrete and independent modules that are assumed to be capable of doing

their respective tasks on their own. Some of these modules may serve as the foundation for the

entire software system. Generally speaking, designers create modules in such a way that they

can be executed and/or compiled independently of one another.

When it comes to problem-solving strategies, modular design accidentally follows the rules of

the 'split and conquer' technique. This is due to the fact that the modular design of software

has a plethora of additional advantages.

Modularization has the following advantages:

* Smaller components are less difficult to keep up to date.
* The program can be separated into sections based on its functional elements.
* The software can be configured to operate at the desired level of abstraction.
* Components having a high degree of cohesiveness can be reused multiple times.
* It is feasible to implement concurrent execution.

It is a measure of the degree of infra-dependability between pieces of a module that is used to

describe cohesion. The greater the degree of cohesion, the better the program's overall design.

There are seven forms of cohesiveness, which are as follows:

Co-incidental cohesiveness is unplanned and random cohesion that can occur as a result of the

programme being broken down into smaller modules for the sake of modularization, for example. Because it was unexpected, it may generate misunderstanding among programmers, and it is therefore not usually accepted.

Logical cohesion is the term used to describe the process of grouping elements that have been

logically classified into a single module.

A concept known as temporal cohesiveness refers to the organisation of module pieces so that

they are processed at roughly the same time in a given module.

Cohesiveness at functional level - This is believed to be the highest level of cohesion and is

greatly anticipated. Because they all contribute to a single defined function, elements of

modules in functional cohesion are put together to form a module.

**CODING**

**appControl.py**

import pyscreenshot as ImageGrab

import time

import os

import subprocess

from pynput.keyboard import Key, Controller

import psutil

class SystemTasks:

    def \_\_init\_\_(self):

        self.keyboard = Controller()

    def openApp(self, appName):

        appName = appName.replace('paint', 'mspaint')

        appName = appName.replace('wordpad', 'write')

        appName = appName.replace('word', 'write')

        appName = appName.replace('calculator', 'calc')

        try: subprocess.Popen('C:\\Windows\\System32\\'+appName[5:]+'.exe')

        except: pass

    def write(self, text):

        text = text[5:]

        for char in text:

            self.keyboard.type(char)

            time.sleep(0.02)

    def select(self):

        self.keyboard.press(Key.ctrl)

        self.keyboard.press('a')

        self.keyboard.release('a')

        self.keyboard.release(Key.ctrl)

    def hitEnter(self):

        self.keyboard.press(Key.enter)

        self.keyboard.release(Key.enter)

    def delete(self):

        self.keyboard.press(Key.backspace)

        self.keyboard.release(Key.enter)

    def save(self, text):

        if "don't" in text:

            self.keyboard.press(Key.right)

        else:

            self.keyboard.press(Key.ctrl)

            self.keyboard.press('s')

            self.keyboard.release('s')

            self.keyboard.release(Key.ctrl)

        self.hitEnter()

class TabOpt:

    def \_\_init\_\_(self):

        self.keyboard = Controller()

    def switchTab(self):

        self.keyboard.press(Key.ctrl)

        self.keyboard.press(Key.tab)

        self.keyboard.release(Key.tab)

        self.keyboard.release(Key.ctrl)

    def closeTab(self):

        self.keyboard.press(Key.ctrl)

        self.keyboard.press('w')

        self.keyboard.release('w')

        self.keyboard.release(Key.ctrl)

    def newTab(self):

        self.keyboard.press(Key.ctrl)

        self.keyboard.press('n')

        self.keyboard.release('n')

        self.keyboard.release(Key.ctrl)

class WindowOpt:

    def \_\_init\_\_(self):

        self.keyboard = Controller()

    def openWindow(self):

        self.maximizeWindow()

    def closeWindow(self):

        self.keyboard.press(Key.alt\_l)

        self.keyboard.press(Key.f4)

        self.keyboard.release(Key.f4)

        self.keyboard.release(Key.alt\_l)

    def minimizeWindow(self):

        for i in range(2):

            self.keyboard.press(Key.cmd)

            self.keyboard.press(Key.down)

            self.keyboard.release(Key.down)

            self.keyboard.release(Key.cmd)

            time.sleep(0.05)

    def maximizeWindow(self):

        self.keyboard.press(Key.cmd)

        self.keyboard.press(Key.up)

        self.keyboard.release(Key.up)

        self.keyboard.release(Key.cmd)

    def moveWindow(self, operation):

        self.keyboard.press(Key.cmd)

        if "left" in operation:

            self.keyboard.press(Key.left)

            self.keyboard.release(Key.left)

        elif "right" in operation:

            self.keyboard.press(Key.right)

            self.keyboard.release(Key.right)

        elif "down" in operation:

            self.keyboard.press(Key.down)

            self.keyboard.release(Key.down)

        elif "up" in operation:

            self.keyboard.press(Key.up)

            self.keyboard.release(Key.up)

        self.keyboard.release(Key.cmd)

    def switchWindow(self):

        self.keyboard.press(Key.alt\_l)

        self.keyboard.press(Key.tab)

        self.keyboard.release(Key.tab)

        self.keyboard.release(Key.alt\_l)

    def takeScreenShot(self):

        from random import randint

        im = ImageGrab.grab()

        im.save(f'Files and Document/ss\_{randint(1, 100)}.jpg')

def isContain(text, lst):

    for word in lst:

        if word in text:

            return True

    return False

def Win\_Opt(operation):

    w = WindowOpt()

    if isContain(operation, ['open']):

        w.openWindow()

    elif isContain(operation, ['close']):

        w.closeWindow()

    elif isContain(operation, ['mini']):

        w.minimizeWindow()

    elif isContain(operation, ['maxi']):

        w.maximizeWindow()

    elif isContain(operation, ['move', 'slide']):

        w.moveWindow(operation)

    elif isContain(operation, ['switch','which']):

        w.switchWindow()

    elif isContain(operation, ['screenshot','capture','snapshot']):

        w.takeScreenShot()

    return

def Tab\_Opt(operation):

    t = TabOpt()

    if isContain(operation, ['new','open','another','create']):

        t.newTab()

    elif isContain(operation, ['switch','move','another','next','previous','which']):

        t.switchTab()

    elif isContain(operation, ['close','delete']):

        t.closeTab()

    else:

        return

def System\_Opt(operation):

    s = SystemTasks()

    if 'delete' in operation:

        s.delete()

    elif 'save' in operation:

        s.save(operation)

    elif 'type' in operation:

        s.write(operation)

    elif 'select' in operation:

        s.select()

    elif 'enter' in operation:

        s.hitEnter()

    elif isContain(operation, ['notepad','paint','calc','word']):

        s.openApp(operation)

    elif isContain(operation, ['music','video']):

        s.playMusic(operation)

    else:

        open\_website(operation)

        return

###########  VOLUME ###########

keyboard = Controller()

def mute():

    for i in range(50):

        keyboard.press(Key.media\_volume\_down)

        keyboard.release(Key.media\_volume\_down)

def full():

    for i in range(50):

        keyboard.press(Key.media\_volume\_up)

        keyboard.release(Key.media\_volume\_up)

def volumeControl(text):

    if 'full' in text or 'max' in text: full()

    elif 'mute' in text or 'min' in text: mute()

    elif 'incre' in text:

        for i in range(5):

            keyboard.press(Key.media\_volume\_up)

            keyboard.release(Key.media\_volume\_up)

    elif 'decre' in text:

        for i in range(5):

            keyboard.press(Key.media\_volume\_down)

            keyboard.release(Key.media\_volume\_down)

def systemInfo():

    import wmi

    c = wmi.WMI()

    my\_system\_1 = c.Win32\_LogicalDisk()[0]

    my\_system\_2 = c.Win32\_ComputerSystem()[0]

    info = ["Total Disk Space: " + str(round(int(my\_system\_1.Size)/(1024\*\*3),2)) + " GB",

            "Free Disk Space: " + str(round(int(my\_system\_1.Freespace)/(1024\*\*3),2)) + " GB",

            "Manufacturer: " + my\_system\_2.Manufacturer,

            "Model: " + my\_system\_2. Model,

            "Owner: " + my\_system\_2.PrimaryOwnerName,

            "Number of Processors: " + str(my\_system\_2.NumberOfProcessors),

            "System Type: " + my\_system\_2.SystemType]

    return info

def batteryInfo():

    battery = psutil.sensors\_battery()

    pr = str(battery.percent)

    if battery.power\_plugged:

        return "Your System is currently on Charging Mode and it's " + pr + "% done."

    return "Your System is currently on " + pr + "% battery life."

def OSHandler(query):

    if isContain(query, ['system', 'info']):

        return ['Here is your System Information...', '\n'.join(systemInfo())]

    elif isContain(query, ['cpu', 'battery']):

        return batteryInfo()

from difflib import get\_close\_matches

import json

from random import choice

import webbrowser

data = json.load(open('extrafiles/websites.json', encoding='utf-8'))

def open\_website(query):

    query = query.replace('open','')

    if query in data:

        response = data[query]

    else:

        query = get\_close\_matches(query, data.keys(), n=2, cutoff=0.5)

        if len(query)==0: return "None"

        response = choice(data[query[0]])

    webbrowser.open(response)

**ChooseAvatarPIC.py**

from tkinter import \*

from PIL import Image, ImageTk

from tkinter import ttk

from pynput.keyboard import Key, Controller

import userHandler

from userHandler import UserData

u = UserData()

u.extractData()

avatarChoosen = u.getUserPhoto()

def closeWindow():

    keyboard = Controller()

    keyboard.press(Key.alt\_l)

    keyboard.press(Key.f4)

    keyboard.release(Key.f4)

    keyboard.release(Key.alt\_l)

def SavePhoto():

    userHandler.UpdateUserPhoto(avatarChoosen)

    closeWindow()

def selectAVATAR(avt=0):

    global avatarChoosen

    avatarChoosen = avt

    i=1

    for avtr in (avtb1,avtb2,avtb3,avtb4,avtb5,avtb6,avtb7,avtb8,avtb9,avtb10,avtb11,avtb12,avtb13,avtb14,avtb15):

        if i==avt:

            avtr['state'] = 'disabled'

        else:

            avtr['state'] = 'normal'

        i+=1

if \_\_name\_\_ == "\_\_main\_\_":

    background = '#F6FAFB'

    avtrRoot = Tk()

    avtrRoot.title("Choose Avatar")

    avtrRoot.configure(bg=background)

    w\_width, w\_height = 500, 450

    s\_width, s\_height = avtrRoot.winfo\_screenwidth(), avtrRoot.winfo\_screenheight()

    x, y = (s\_width/2)-(w\_width/2), (s\_height/2)-(w\_height/2)

    avtrRoot.geometry('%dx%d+%d+%d' % (w\_width,w\_height,x,y-30))

    Label(avtrRoot, text="Choose Your Avatar", font=('arial bold', 15), bg=background, fg='#303E54').pack(pady=10)

    avatarContainer = Frame(avtrRoot, bg=background)

    avatarContainer.pack(pady=10, ipadx=50, ipady=20)

    size = 100

    main\_frame = Frame(avatarContainer)

    main\_frame.pack(fill=BOTH, expand=1)

    my\_canvas = Canvas(main\_frame, bg=background)

    my\_canvas.pack(side=LEFT, expand=1, fill=BOTH)

    my\_scrollbar = ttk.Scrollbar(main\_frame, orient=VERTICAL, command=my\_canvas.yview)

    my\_scrollbar.pack(side=RIGHT, fill=Y)

    my\_canvas.configure(yscrollcommand=my\_scrollbar.set)

    my\_canvas.bind('<Configure>', lambda e: my\_canvas.configure(scrollregion = my\_canvas.bbox('all')))

    second\_frame = Frame(my\_canvas)

    my\_canvas.create\_window((0,0), window=second\_frame, anchor='nw')

    avtr1 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a1.png').resize((size, size)), Image.ANTIALIAS)

    avtr2 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a2.png').resize((size, size)), Image.ANTIALIAS)

    avtr3 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a3.png').resize((size, size)), Image.ANTIALIAS)

    avtr4 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a4.png').resize((size, size)), Image.ANTIALIAS)

    avtr5 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a5.png').resize((size, size)), Image.ANTIALIAS)

    avtr6 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a6.png').resize((size, size)), Image.ANTIALIAS)

    avtr7 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a7.png').resize((size, size)), Image.ANTIALIAS)

    avtr8 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a8.png').resize((size, size)), Image.ANTIALIAS)

    avtr9 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a9.png').resize((size, size)), Image.ANTIALIAS)

    avtr10 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a10.png').resize((size, size)), Image.ANTIALIAS)

    avtr11 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a11.png').resize((size, size)), Image.ANTIALIAS)

    avtr12 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a12.png').resize((size, size)), Image.ANTIALIAS)

    avtr13 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a13.png').resize((size, size)), Image.ANTIALIAS)

    avtr14 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a14.png').resize((size, size)), Image.ANTIALIAS)

    avtr15 = ImageTk.PhotoImage(Image.open('extrafiles/images/avatars/a15.png').resize((size, size)), Image.ANTIALIAS)

    avtb1 = Button(second\_frame, image=avtr1, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(1))

    avtb2 = Button(second\_frame, image=avtr2, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(2))

    avtb3 = Button(second\_frame, image=avtr3, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(3))

    avtb4 = Button(second\_frame, image=avtr4, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(4))

    avtb5 = Button(second\_frame, image=avtr5, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(5))

    avtb6 = Button(second\_frame, image=avtr6, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(6))

    avtb7 = Button(second\_frame, image=avtr7, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(7))

    avtb8 = Button(second\_frame, image=avtr8, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(8))

    avtb9 = Button(second\_frame, image=avtr9, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(9))

    avtb10 = Button(second\_frame, image=avtr10, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(10))

    avtb11 = Button(second\_frame, image=avtr11, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(11))

    avtb12 = Button(second\_frame, image=avtr12, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(12))

    avtb13 = Button(second\_frame, image=avtr13, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(13))

    avtb14 = Button(second\_frame, image=avtr14, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(14))

    avtb15 = Button(second\_frame, image=avtr15, bg=background, activebackground=background, relief=FLAT, bd=0, command=lambda:selectAVATAR(15))

    avtb1.grid( row=0, column=0, ipadx=25, ipady=10)

    avtb2.grid( row=0, column=1, ipadx=25, ipady=10)

    avtb3.grid( row=0, column=2, ipadx=25, ipady=10)

    avtb4.grid( row=1, column=0, ipadx=25, ipady=10)

    avtb5.grid( row=1, column=1, ipadx=25, ipady=10)

    avtb6.grid( row=1, column=2, ipadx=25, ipady=10)

    avtb7.grid( row=2, column=0, ipadx=25, ipady=10)

    avtb8.grid( row=2, column=1, ipadx=25, ipady=10)

    avtb9.grid( row=2, column=2, ipadx=25, ipady=10)

    avtb10.grid(row=3, column=0, ipadx=25, ipady=10)

    avtb11.grid(row=3, column=1, ipadx=25, ipady=10)

    avtb12.grid(row=3, column=2, ipadx=25, ipady=10)

    avtb13.grid(row=4, column=0, ipadx=25, ipady=10)

    avtb14.grid(row=4, column=1, ipadx=25, ipady=10)

    avtb15.grid(row=4, column=2, ipadx=25, ipady=10)

    BottomFrame = Frame(avtrRoot, bg=background)

    BottomFrame.pack(pady=10)

    Button(BottomFrame, text='         Update         ', font=('Montserrat Bold', 15), bg='#01933B', fg='white', bd=0, relief=FLAT, command=SavePhoto).grid(row=0, column=0, padx=10)

    Button(BottomFrame, text='         Cancel         ', font=('Montserrat Bold', 15), bg='#EDEDED', fg='#3A3834', bd=0, relief=FLAT, command=closeWindow).grid(row=0, column=1, padx=10)

    avtrRoot.iconbitmap("extrafiles/images/changeProfile.ico")

    avtrRoot.mainloop()

**Dictionary.py**

from difflib import get\_close\_matches

import json

from random import choice

data = json.load(open('extrafiles/dict\_data.json', encoding='utf-8'))

def getMeaning(word):

    if word in data:

        return word, data[word], 1

    elif len(get\_close\_matches(word, data.keys())) > 0:

        word = get\_close\_matches(word, data.keys())[0]

        return word, data[word], 0

    else:

        return word, ["This word doesn't exists in the dictionary."], -1

def translate(query):

    query = query.replace('dictionary', '')

    if 'meaning' in query:

        ind = query.index('meaning of')

        word = query[ind+10:].strip().lower()

    elif 'definition' in query:

        try:

            ind = query.index('definition of')

            word = query[ind+13:].strip().lower()

        except:

            ind = query.index('definition')

            word = query[ind+10:].strip().lower()

    else: word = query

    word, result, check = getMeaning(word)

    result = choice(result)

    if check==1:

        return ["Here's the definition of \"" +word.capitalize()+ '"', result]

    elif check==0:

        return ["I think you're looking for \"" +word.capitalize()+ '"', "It's definition is,\n" + result]

    else:

        return [result, '']

**Filehandler.py**

import subprocess

import wmi

import os

import sys

import webbrowser

if os.path.exists('Files and Document') == False:

    os.mkdir('Files and Document')

path = 'Files and Document/'

def isContain(text, list):

    for word in list:

        if word in text:

            return True

    return False

def createFile(text):

    appLocation = "C:\\Program Files\\Sublime Text 3\\sublime\_text.exe"

    if isContain(text, ["ppt","power point","powerpoint"]):

        file\_name = "sample\_file.ppt"

        appLocation = "C:\\Program Files (x86)\\Microsoft Office\\Office15\\POWERPNT.exe"

    elif isContain(text, ['excel','spreadsheet']):

        file\_name = "sample\_file.xsl"

        appLocation = "C:\\Program Files (x86)\\Microsoft Office\\Office15\\EXCEL.EXE"

    elif isContain(text, ['word','document']):

        file\_name = "sample\_file.docx"

        appLocation = "C:\\Program Files (x86)\\Microsoft Office\\Office15\\WINWORD.EXE"

    elif isContain(text, ["text","simple","normal"]): file\_name = "sample\_file.txt"

    elif "python" in text: file\_name = "sample\_file.py"

    elif "css" in text: file\_name = "sample\_file.css"

    elif "javascript" in text: file\_name = "sample\_file.js"

    elif "html" in text: file\_name = "sample\_file.html"

    elif "c plus plus" in text or "c + +" in text: file\_name = "sample\_file.cpp"

    elif "java" in text: file\_name = "sample\_file.java"

    elif "json" in text: file\_name = "sample\_file.json"

    else: return "Unable to create this type of file"

    file = open(path + file\_name, 'w')

    file.close()

    subprocess.Popen([appLocation, path + file\_name])

    return "File is created.\nNow you can edit this file"

def CreateHTMLProject(project\_name='Sample'):

    if os.path.isdir(path + project\_name):

        webbrowser.open(os.getcwd() + '/' + path + project\_name + "\\index.html")

        return 'There is a same project which is already created, look at this...'

    else:

        os.mkdir(path + project\_name)

    os.mkdir(path+project\_name+ '/images')

    os.mkdir(path+project\_name+ '/videos')

    htmlContent = '<html>\n\t<head>\n\t\t<title> ' + project\_name + ' </title>\n\t\t<link rel="stylesheet" type="text/css" href="style.css">\n\t</head>\n<body>\n\t<p id="label"></p>\n\t<button id="btn" onclick="showText()"> Click Me </button>\n\t<script src="script.js"></script>\n</body>\n</html>'

    htmlFile = open(path+project\_name+ '/index.html', 'w')

    htmlFile.write(htmlContent)

    htmlFile.close()

    cssContent = '\* {\n\tmargin:0;\n\tpadding:0;\n}\nbody {\n\theight:100vh;\n\tdisplay:flex;\n\tjustify-content:center;\n\talign-items:center;\n}\n#btn {\n\twidth:200px;\n\tpadding: 20px 10px;\n\tborder-radius:5px;\n\tbackground-color:red;\n\tcolor:#fff;\n\toutline:none;border:none;\n}\np {\n\tfont-size:30px;\n}'

    cssFile = open(path+project\_name+ '/style.css', 'w')

    cssFile.write(cssContent)

    cssFile.close

    jsContent = 'function showText() {\n\tdocument.getElementById("label").innerHTML="Successfully Created '+ project\_name +' Project";\n\tdocument.getElementById("btn").style="background-color:green;"\n}'

    jsFile = open(path+project\_name+ '/script.js', 'w')

    jsFile.write(jsContent)

    jsFile.close()

    appLocation = "C:\\Program Files\\Sublime Text 3\\sublime\_text.exe"

    # subprocess.Popen([appLocation, path + project\_name])

    subprocess.Popen([appLocation, path + project\_name + "/index.html"])

    subprocess.Popen([appLocation, path + project\_name + "/style.css"])

    subprocess.Popen([appLocation, path + project\_name + "/script.js"])

    webbrowser.open(os.getcwd() + '/' + path + project\_name + "\\index.html")

    return f'Successfully Created {project\_name} Project'

**Game.py**

from random import \*

import playsound

from tkinter import \*

from PIL import Image, ImageTk

from threading import Thread

import speech\_recognition as sr

import pyttsx3

import time

from pynput.keyboard import Key, Controller

def closeWindow():

    keyboard = Controller()

    keyboard.press(Key.alt\_l)

    keyboard.press(Key.f4)

    keyboard.release(Key.f4)

    keyboard.release(Key.alt\_l)

try:

    engine = pyttsx3.init()

    voices = engine.getProperty('voices')

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

    engine.setProperty('volume', 1)

except Exception as e:

    print(e)

def speak(text):

    print(text)

    engine.say(text)

    engine.runAndWait()

def record():

    global userchat

    userchat['text'] = "Listening..."

    r = sr.Recognizer()

    r.dynamic\_energy\_threshold = False

    r.energy\_threshold = 4000

    with sr.Microphone() as source:

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

        audio = r.listen(source)

        said = ""

        try:

            said = r.recognize\_google(audio)

            print(f"\nUser said: {said}")

        except Exception as e:

            print(e)

            speak("I think it is invalid move...")

            return "None"

    return said.lower()

moves = ['rock', 'paper', 'scissor']

class RockPaperScissor:

    def \_\_init\_\_(self):

        self.playerScore = 0

        self.botScore = 0

        self.total\_moves = 0

        self.intro()

    def intro(self):

        speak("Welcome to the Rock Paper Scissor Game. To STOP the Match, say STOP or Cancel. Let's Play.")

    def nextMove(self, move):

        global userchat, botchat, totalLabel, botMoveLBL

        userchat['text'] = move.upper()

        botMove = randint(0,2)

        playerMove = moves.index(move)

        botchat['text'] = moves[botMove].upper()

        self.total\_moves += 1

        if botMove==playerMove:

            self.botScore += 1

            self.playerScore += 1

        elif botMove==0:

            if playerMove==1:

                self.playerScore += 1

            else:

                self.botScore += 1

        elif botMove==1:

            if playerMove==2:

                self.playerScore += 1

            else:

                self.botScore += 1

        else:

            if playerMove==0:

                self.playerScore += 1

            else:

                self.botScore += 1

        totalLabel['text'] = str(self.botScore)+'   |   '+str(self.playerScore)

        if botMove==0: botMoveLBL['image'] = rockImg

        if botMove==1: botMoveLBL['image'] = paperImg

        if botMove==2: botMoveLBL['image'] = scissorImg

        speak('I choose: ' + str(moves[botMove]))

        return botMove+1

    def whoWon(self):

        result = ""

        if self.playerScore == self.botScore:

            result = "The match is draw !\n"

        elif self.playerScore > self.botScore:

            result = "You won the match Sir! Well Done !\n"

        else:

            result = "You lose the match Sir! Haha!\n"

        for el in root.winfo\_children():

            el.destroy()

        if 'won' in result:

            Label(root, image=winImg).pack(pady=30)

        elif 'lose' in result:

            Label(root, image=loseImg).pack(pady=30)

        else:

            Label(root, image=drawImg).pack(pady=30)

        result += "You have won " +str(self.playerScore)+"/"+str(self.total\_moves)+" matches."

        Label(root, text='Score', font=('Arial Bold', 50), fg='#FE8A28', bg='white').pack()

        Label(root, text=str(self.playerScore)+' / '+str(self.total\_moves), font=('Arial Bold', 40), fg='#292D3E', bg='white').pack()

        speak(result)

        time.sleep(1)

        closeWindow()

        return

rockImg, paperImg, scissorImg, userchat, botchat, totalLabel, botMoveLBL, userMoveLBL, winImg, loseImg, drawImg = None, None, None, None, None, None, None, None, None, None, None

def playRock():

    rp = RockPaperScissor()

    while True:

        global botMoveLBL, userMoveLBL

        move = record()

        if isContain(move, ["don't", "cancel", "stop"]):

            rp.whoWon()

            break

        else:

            img = None

            if 'rock' in move:

                userMoveLBL['image'] = rockImg

                img = rp.nextMove('rock')

            elif 'paper' in move:

                userMoveLBL['image'] = paperImg

                img = rp.nextMove('paper')

            elif 'scissor' in move or 'caesar' in move:

                userMoveLBL['image'] = scissorImg

                img = rp.nextMove('scissor')

def rockPaperScissorWindow():

    global root, rockImg, paperImg, scissorImg, userchat, botchat, totalLabel, botMoveLBL, userMoveLBL, winImg, loseImg, drawImg

    root = Tk()

    root.title('Rock Paper Scissor')

    # root.resizable(0,0)

    # root.attributes('-toolwindow', True)

    w\_width, w\_height = 400, 650

    s\_width, s\_height = root.winfo\_screenwidth(), root.winfo\_screenheight()

    x, y = (s\_width/2)-(w\_width/2), (s\_height/2)-(w\_height/2)

    root.geometry('%dx%d+%d+%d' % (w\_width,w\_height,x,y-30)) #center location of the screen

    root.configure(bg='white')

    rockImg = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/1.jpg'))

    paperImg = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/2.jpg'))

    scissorImg = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/3.jpg'))

    grayImg = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/grayQuestion.png'))

    orangeImg = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/orangeQuestion.jpg'))

    winImg = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/win.jpg'))

    loseImg = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/lose.jpg'))

    drawImg = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/draw.jpg'))

    toplbl = Label(root, text='Total Score', font=('Arial Bold', 20), fg='#FE8A28', bg='white').pack()

    totalLabel = Label(root, text='0   |   0', font=('Arial Bold', 15), fg='#1F1F1F', bg='white')

    totalLabel.pack()

    #bottom image

    img = ImageTk.PhotoImage(Image.open('extrafiles/ROCKPAPERSCISSOR/rockPaperScissor.jpg'))

    downLbl = Label(root, image=img)

    downLbl.pack(side=BOTTOM)

    #user response

    userchat = Label(root, text='Listening...', bg='#FE8A28', fg='white', font=('Arial Bold',13))

    userchat.place(x=300, y=120)

    userMoveLBL = Label(root, image=orangeImg)

    userMoveLBL.place(x=260, y=150)

    #bot response

    botchat = Label(root, text='Waiting...', bg='#EAEAEA', fg='#494949', font=('Arial Bold',13))

    botchat.place(x=12, y=120)

    botMoveLBL = Label(root, image=grayImg)

    botMoveLBL.place(x=12, y=150)

    Thread(target=playRock).start()

    root.iconbitmap("extrafiles/images/game.ico")

    root.mainloop()

def isContain(text, lst):

    for word in lst:

        if word in text:

            return True

    return False

def play(gameName):

    speak('')

    if isContain(gameName, ['dice','die']):

        playsound.playsound('extrafiles/audios/dice.mp3')

        result = "You got " + str(randint(1,6))

        return result

    elif isContain(gameName, ['coin']):

        playsound.playsound('extrafiles/audios/coin.mp3')

        p = randint(-10,10)

        if p>0: return "You got Head"

        else: return "You got Tail"

    elif isContain(gameName, ['rock','paper','scissor','first']):

        rockPaperScissorWindow()

        return

    else:

        print("Game Not Available")

def showGames():

    return "1. Rock Paper Scissor\n2. Online Games"

**GUIASSISTANT.py**

# GLOBAL VARIABLES USED #

ai\_name = 'Personal Assistant'.lower()

EXIT\_COMMANDS = ['bye','exit','quit','shut down', 'shutdown']

ownerName = "Parth"

ownerDesignation = "Sir"

ownerPhoto = "1"

rec\_email, rec\_phoneno = "", ""

WAEMEntry = None

avatarChoosen = 0

choosedAvtrImage = None

botChatTextBg = "#007cc7"

botChatText = "white"

userChatTextBg = "#4da8da"

chatBgColor = '#12232e'

background = '#203647'

textColor = 'white'

AITaskStatusLblBG = '#203647'

KCS\_IMG = 1 #0 for light, 1 for dark

voice\_id = 0 #0 for female, 1 for male

ass\_volume = 1 #max volume

ass\_voiceRate = 200 #normal voice rate

####################################### IMPORTING MODULES ###########################################

""" User Created Modules """

try:

    import normalChat

    import math\_function

    import appControl

    import webScrapping

    import game

    from userHandler import UserData

    import timer

    import dictionary

    import ToDo

    import fileHandler

except Exception as e:

    raise e

""" System Modules """

try:

    import os

    import speech\_recognition as sr

    import pyttsx3

    from tkinter import \*

    from tkinter import ttk

    from tkinter import messagebox

    from tkinter import colorchooser

    from PIL import Image, ImageTk

    from time import sleep

    from threading import Thread

except Exception as e:

    print(e)

if os.path.exists('userData')==False:

    os.mkdir('userData')

########################################## BOOT UP WINDOW ###########################################

def ChangeSettings(write=False):

    import pickle

    global background, textColor, chatBgColor, voice\_id, ass\_volume, ass\_voiceRate, AITaskStatusLblBG, KCS\_IMG, botChatTextBg, botChatText, userChatTextBg

    setting = {'background': background,

                'textColor': textColor,

                'chatBgColor': chatBgColor,

                'AITaskStatusLblBG': AITaskStatusLblBG,

                'KCS\_IMG': KCS\_IMG,

                'botChatText': botChatText,

                'botChatTextBg': botChatTextBg,

                'userChatTextBg': userChatTextBg,

                'voice\_id': voice\_id,

                'ass\_volume': ass\_volume,

                'ass\_voiceRate': ass\_voiceRate

            }

    if write:

        with open('userData/settings.pck', 'wb') as file:

            pickle.dump(setting, file)

        return

    try:

        with open('userData/settings.pck', 'rb') as file:

            loadSettings = pickle.load(file)

            background = loadSettings['background']

            textColor = loadSettings['textColor']

            chatBgColor = loadSettings['chatBgColor']

            AITaskStatusLblBG = loadSettings['AITaskStatusLblBG']

            KCS\_IMG = loadSettings['KCS\_IMG']

            botChatText = loadSettings['botChatText']

            botChatTextBg = loadSettings['botChatTextBg']

            userChatTextBg = loadSettings['userChatTextBg']

            voice\_id = loadSettings['voice\_id']

            ass\_volume = loadSettings['ass\_volume']

            ass\_voiceRate = loadSettings['ass\_voiceRate']

    except Exception as e:

        pass

if os.path.exists('userData/settings.pck')==False:

    ChangeSettings(True)

def getChatColor():

    global chatBgColor

    chatBgColor = myColor[1]

    colorbar['bg'] = chatBgColor

    chat\_frame['bg'] = chatBgColor

    root1['bg'] = chatBgColor

def changeTheme():

    global background, textColor, AITaskStatusLblBG, KCS\_IMG, botChatText, botChatTextBg, userChatTextBg, chatBgColor

    if themeValue.get()==1:

        background, textColor, AITaskStatusLblBG, KCS\_IMG = "#203647", "white", "#203647",1

        cbl['image'] = cblDarkImg

        kbBtn['image'] = kbphDark

        settingBtn['image'] = sphDark

        AITaskStatusLbl['bg'] = AITaskStatusLblBG

        botChatText, botChatTextBg, userChatTextBg = "white", "#007cc7", "#4da8da"

        chatBgColor = "#12232e"

        colorbar['bg'] = chatBgColor

    else:

        background, textColor, AITaskStatusLblBG, KCS\_IMG = "#F6FAFB", "#303E54", "#14A769", 0

        cbl['image'] = cblLightImg

        kbBtn['image'] = kbphLight

        settingBtn['image'] = sphLight

        AITaskStatusLbl['bg'] = AITaskStatusLblBG

        botChatText, botChatTextBg, userChatTextBg = "#494949", "#EAEAEA", "#23AE79"

        chatBgColor = "#F6FAFB"

        colorbar['bg'] = '#E8EBEF'

    root['bg'], root2['bg'] = background, background

    settingsFrame['bg'] = background

    settingsLbl['fg'], userPhoto['fg'], userName['fg'], assLbl['fg'], voiceRateLbl['fg'], volumeLbl['fg'], themeLbl['fg'], chooseChatLbl['fg'] = textColor, textColor, textColor, textColor, textColor, textColor, textColor, textColor

    settingsLbl['bg'], userPhoto['bg'], userName['bg'], assLbl['bg'], voiceRateLbl['bg'], volumeLbl['bg'], themeLbl['bg'], chooseChatLbl['bg'] = background, background, background, background, background, background, background, background

    s.configure('Wild.TRadiobutton', background=background, foreground=textColor)

    volumeBar['bg'], volumeBar['fg'], volumeBar['highlightbackground'] = background, textColor, background

    chat\_frame['bg'], root1['bg'] = chatBgColor, chatBgColor

    userPhoto['activebackground'] = background

    ChangeSettings(True)

def changeVoice(e):

    global voice\_id

    voice\_id=0

    if assVoiceOption.get()=='Male': voice\_id=1

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

    ChangeSettings(True)

def changeVolume(e):

    global ass\_volume

    ass\_volume = volumeBar.get() / 100

    engine.setProperty('volume', ass\_volume)

    ChangeSettings(True)

def changeVoiceRate(e):

    global ass\_voiceRate

    temp = voiceOption.get()

    if temp=='Very Low': ass\_voiceRate = 100

    elif temp=='Low': ass\_voiceRate = 150

    elif temp=='Fast': ass\_voiceRate = 250

    elif temp=='Very Fast': ass\_voiceRate = 300

    else: ass\_voiceRate = 200

    print(ass\_voiceRate)

    engine.setProperty('rate', ass\_voiceRate)

    ChangeSettings(True)

ChangeSettings()

############################################ SET UP VOICE ###########################################

try:

    engine = pyttsx3.init()

    voices = engine.getProperty('voices')

    engine.setProperty('voice', voices[voice\_id].id) #male

    engine.setProperty('volume', ass\_volume)

except Exception as e:

    print(e)

####################################### SET UP TEXT TO SPEECH #######################################

def speak(text, display=False, icon=False):

    AITaskStatusLbl['text'] = 'Speaking...'

    if icon: Label(chat\_frame, image=botIcon, bg=chatBgColor).pack(anchor='w',pady=0)

    if display: attachTOframe(text, True)

    print('\n'+ai\_name.upper()+': '+text)

    try:

        engine.say(text)

        engine.runAndWait()

    except:

        print("Try not to type more...")

####################################### SET UP SPEECH TO TEXT #######################################

def record(clearChat=True, iconDisplay=True):

    print('\nListening...')

    AITaskStatusLbl['text'] = 'Listening...'

    r = sr.Recognizer()

    r.dynamic\_energy\_threshold = False

    r.energy\_threshold = 4000

    with sr.Microphone() as source:

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

        audio = r.listen(source)

        said = ""

        try:

            AITaskStatusLbl['text'] = 'Processing...'

            said = r.recognize\_google(audio)

            print(f"\nUser said: {said}")

            if clearChat:

                clearChatScreen()

            if iconDisplay: Label(chat\_frame, image=userIcon, bg=chatBgColor).pack(anchor='e',pady=0)

            attachTOframe(said)

        except Exception as e:

            print(e)

            # speak("I didn't get it, Say that again please...")

            if "connection failed" in str(e):

                speak("Your System is Offline...", True, True)

            return 'None'

    return said.lower()

def voiceMedium():

    while True:

        query = record()

        if query == 'None': continue

        if isContain(query, EXIT\_COMMANDS):

            speak("Shutting down the System. Good Bye "+ownerDesignation+"!", True, True)

            break

        else: main(query.lower())

    appControl.Win\_Opt('close')

def keyboardInput(e):

    user\_input = UserField.get().lower()

    if user\_input!="":

        clearChatScreen()

        if isContain(user\_input, EXIT\_COMMANDS):

            speak("Shutting down the System. Good Bye "+ownerDesignation+"!", True, True)

        else:

            Label(chat\_frame, image=userIcon, bg=chatBgColor).pack(anchor='e',pady=0)

            attachTOframe(user\_input.capitalize())

            Thread(target=main, args=(user\_input,)).start()

        UserField.delete(0, END)

###################################### TASK/COMMAND HANDLER #########################################

def isContain(txt, lst):

    for word in lst:

        if word in txt:

            return True

    return False

def main(text):

        if "project" in text:

            if isContain(text, ['make', 'create']):

                speak("What do you want to give the project name ?", True, True)

                projectName = record(False, False)

                speak(fileHandler.CreateHTMLProject(projectName.capitalize()), True)

                return

        if "create" in text and "file" in text:

            speak(fileHandler.createFile(text), True, True)

            return

        if "translate" in text:

            speak("What do you want to translate?", True, True)

            sentence = record(False, False)

            speak("Which langauage to translate ?", True)

            langauage = record(False, False)

            result = normalChat.lang\_translate(sentence, langauage)

            if result=="None": speak("This langauage doesn't exists")

            else:

                speak(f"In {langauage.capitalize()} you would say:", True)

                if langauage=="hindi":

                    attachTOframe(result.text, True)

                    speak(result.pronunciation)

                else: speak(result.text, True)

            return

        if 'list' in text:

            if isContain(text, ['add', 'create', 'make']):

                speak("What do you want to add?", True, True)

                item = record(False, False)

                ToDo.toDoList(item)

                speak("Alright, I added to your list", True)

                return

            if isContain(text, ['show', 'my list']):

                items = ToDo.showtoDoList()

                if len(items)==1:

                    speak(items[0], True, True)

                    return

                attachTOframe('\n'.join(items), True)

                speak(items[0])

                return

        if isContain(text, ['battery', 'system info']):

            result = appControl.OSHandler(text)

            if len(result)==2:

                speak(result[0], True, True)

                attachTOframe(result[1], True)

            else:

                speak(result, True, True)

            return

        if isContain(text, ['meaning', 'dictionary', 'definition', 'define']):

            result = dictionary.translate(text)

            speak(result[0], True, True)

            if result[1]=='': return

            speak(result[1], True)

            return

        if 'volume' in text:

            appControl.volumeControl(text)

            Label(chat\_frame, image=botIcon, bg=chatBgColor).pack(anchor='w',pady=0)

            attachTOframe('Volume Settings Changed', True)

            return

        if isContain(text, ['timer', 'countdown']):

            Thread(target=timer.startTimer, args=(text,)).start()

            speak('Ok, Timer Started!', True, True)

            return

        if 'whatsapp' in text:

            speak("Sure "+ownerDesignation+"...", True, True)

            speak('Whom do you want to send the message?', True)

            WAEMPOPUP("WhatsApp", "Phone Number")

            attachTOframe(rec\_phoneno)

            speak('What is the message?', True)

            message = record(False, False)

            Thread(target=webScrapping.sendWhatsapp, args=(rec\_phoneno, message,)).start()

            speak("Message is on the way. Do not move away from the screen.")

            attachTOframe("Message Sent", True)

            return

        if 'email' in text:

            speak('Whom do you want to send the email?', True, True)

            WAEMPOPUP("Email", "E-mail Address")

            attachTOframe(rec\_email)

            speak('What is the Subject?', True)

            subject = record(False, False)

            speak('What message you want to send ?', True)

            message = record(False, False)

            Thread(target=webScrapping.email, args=(rec\_email,message,subject,) ).start()

            speak('Email has been Sent', True)

            return

        if isContain(text, ['covid','virus']):

            result = webScrapping.covid(text)

            if 'str' in str(type(result)):

                speak(result, True, True)

                return

            speak(result[0], True, True)

            result = '\n'.join(result[1])

            attachTOframe(result, True)

            return

        if isContain(text, ['youtube','video']):

            speak("Ok "+ownerDesignation+", here a video for you...", True, True)

            try:

                speak(webScrapping.youtube(text), True)

            except Exception as e:

                speak("Desired Result Not Found", True)

            return

        if isContain(text, ['search', 'image']):

            if 'image' in text and 'show' in text:

                Thread(target=showImages, args=(text,)).start()

                speak('Here are the images...', True, True)

                return

            speak(webScrapping.googleSearch(text), True, True)

            return

        if isContain(text, ['map', 'direction']):

            if "direction" in text:

                speak('What is your starting location?', True, True)

                startingPoint = record(False, False)

                speak("Ok "+ownerDesignation+", Where you want to go?", True)

                destinationPoint = record(False, False)

                speak("Ok "+ownerDesignation+", Getting Directions...", True)

                try:

                    distance = webScrapping.giveDirections(startingPoint, destinationPoint)

                    speak('You have to cover a distance of '+ distance, True)

                except:

                    speak("I think location is not proper, Try Again!")

            else:

                webScrapping.maps(text)

                speak('Here you go...', True, True)

            return

        if isContain(text, ['factorial','log','value of','math',' + ',' - ',' x ','multiply','divided by','binary','hexadecimal','octal','shift','sin ','cos ','tan ']):

            try:

                speak(('Result is: ' + math\_function.perform(text)), True, True)

            except Exception as e:

                return

            return

        if "joke" in text:

            speak('Here is a joke...', True, True)

            speak(webScrapping.jokes(), True)

            return

        if isContain(text, ['news']):

            speak('Getting the latest news...', True, True)

            headlines,headlineLinks = webScrapping.latestNews(2)

            for head in headlines: speak(head, True)

            speak('Do you want to read the full news?', True)

            text = record(False, False)

            if isContain(text, ["no","don't"]):

                speak("No Problem "+ownerDesignation, True)

            else:

                speak("Ok "+ownerDesignation+", Opening browser...", True)

                webScrapping.openWebsite('https://indianexpress.com/latest-news/')

                speak("You can now read the full news from this website.")

            return

        if isContain(text, ['weather']):

            data = webScrapping.weather()

            speak('', False, True)

            showSingleImage("weather", data[:-1])

            speak(data[-1])

            return

        if isContain(text, ['screenshot']):

            Thread(target=appControl.Win\_Opt, args=('screenshot',)).start()

            speak("Screen Shot Taken", True, True)

            return

        if isContain(text, ['window','close that']):

            appControl.Win\_Opt(text)

            return

        if isContain(text, ['tab']):

            appControl.Tab\_Opt(text)

            return

        if isContain(text, ['setting']):

            raise\_frame(root2)

            clearChatScreen()

            return

        if isContain(text, ['open','type','save','delete','select','press enter']):

            appControl.System\_Opt(text)

            return

        if isContain(text, ['wiki', 'who is']):

            Thread(target=webScrapping.downloadImage, args=(text, 1,)).start()

            speak('Searching...', True, True)

            result = webScrapping.wikiResult(text)

            showSingleImage('wiki')

            speak(result, True)

            return

        if isContain(text, ['game']):

            speak("Which game do you want to play?", True, True)

            attachTOframe(game.showGames(), True)

            text = record(False)

            if text=="None":

                speak("Didn't understand what you say?", True, True)

                return

            if 'online' in text:

                speak("Ok "+ownerDesignation+", Let's play some online games", True, True)

                webScrapping.openWebsite('https://www.agame.com/games/mini-games/')

                return

            if isContain(text, ["don't", "no", "cancel", "back", "never"]):

                speak("No Problem "+ownerDesignation+", We'll play next time.", True, True)

            else:

                speak("Ok "+ownerDesignation+", Let's Play " + text, True, True)

                os.system(f"python -c \"import game; game.play('{text}')\"")

            return

        if isContain(text, ['coin','dice','die']):

            if "toss" in text or "roll" in text or "flip" in text:

                speak("Ok "+ownerDesignation, True, True)

                result = game.play(text)

                if "Head" in result: showSingleImage('head')

                elif "Tail" in result: showSingleImage('tail')

                else: showSingleImage(result[-1])

                speak(result)

                return

        if isContain(text, ['time','date']):

            speak(normalChat.chat(text), True, True)

            return

        if 'my name' in text:

            speak('Your name is, ' + ownerName, True, True)

            return

        if isContain(text, ['voice']):

            global voice\_id

            try:

                if 'female' in text: voice\_id = 0

                elif 'male' in text: voice\_id = 1

                else:

                    if voice\_id==0: voice\_id=1

                    else: voice\_id=0

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

                ChangeSettings(True)

                speak("Hello "+ownerDesignation+", I have changed my voice. How may I help you?", True, True)

                assVoiceOption.current(voice\_id)

            except Exception as e:

                print(e)

            return

        if isContain(text, ['morning','evening','noon']) and 'good' in text:

            speak(normalChat.chat("good"), True, True)

            return

        result = normalChat.reply(text)

        if result != "None": speak(result, True, True)

        else:

            speak("I couldn't understand your query... ", True, True)

def deleteUserData():

    result = messagebox.askquestion('Alert', 'Are you sure you want to exit ?')

    if result=='no': return

    root.destroy()

                        #####################

                        ####### GUI #########

                        #####################

############ ATTACHING BOT/USER CHAT ON CHAT SCREEN ###########

def attachTOframe(text,bot=False):

    if bot:

        botchat = Label(chat\_frame,text=text, bg=botChatTextBg, fg=botChatText, justify=LEFT, wraplength=250, font=('Montserrat',12, 'bold'))

        botchat.pack(anchor='w',ipadx=5,ipady=5,pady=5)

    else:

        userchat = Label(chat\_frame, text=text, bg=userChatTextBg, fg='white', justify=RIGHT, wraplength=250, font=('Montserrat',12, 'bold'))

        userchat.pack(anchor='e',ipadx=2,ipady=2,pady=5)

def clearChatScreen():

    for wid in chat\_frame.winfo\_children():

        wid.destroy()

### SWITCHING BETWEEN FRAMES ###

def raise\_frame(frame):

    frame.tkraise()

    clearChatScreen()

################# SHOWING DOWNLOADED IMAGES ###############

img0, img1, img2, img3, img4 = None, None, None, None, None

def showSingleImage(type, data=None):

    global img0, img1, img2, img3, img4

    try:

        img0 = ImageTk.PhotoImage(Image.open('Downloads/0.jpg').resize((90,110), Image.ANTIALIAS))

    except:

        pass

    img1 = ImageTk.PhotoImage(Image.open('extrafiles/images/heads.jpg').resize((220,200), Image.ANTIALIAS))

    img2 = ImageTk.PhotoImage(Image.open('extrafiles/images/tails.jpg').resize((220,200), Image.ANTIALIAS))

    img4 = ImageTk.PhotoImage(Image.open('extrafiles/images/WeatherImage.png'))

    if type=="weather":

        weather = Frame(chat\_frame)

        weather.pack(anchor='w')

        Label(weather, image=img4, bg=chatBgColor).pack()

        Label(weather, text=data[0], font=('Arial Bold', 45), fg='white', bg='#3F48CC').place(x=65,y=45)

        Label(weather, text=data[1], font=('Montserrat', 15), fg='white', bg='#3F48CC').place(x=78,y=110)

        Label(weather, text=data[2], font=('Montserrat', 10), fg='white', bg='#3F48CC').place(x=78,y=140)

        Label(weather, text=data[3], font=('Arial Bold', 12), fg='white', bg='#3F48CC').place(x=60,y=160)

    elif type=="wiki":

        Label(chat\_frame, image=img0, bg='#EAEAEA').pack(anchor='w')

    elif type=="head":

        Label(chat\_frame, image=img1, bg='#EAEAEA').pack(anchor='w')

    elif type=="tail":

        Label(chat\_frame, image=img2, bg='#EAEAEA').pack(anchor='w')

    else:

        img3 = ImageTk.PhotoImage(Image.open('extrafiles/images/dice/'+type+'.jpg').resize((200,200), Image.ANTIALIAS))

        Label(chat\_frame, image=img3, bg='#EAEAEA').pack(anchor='w')

def showImages(query):

    global img0, img1, img2, img3

    webScrapping.downloadImage(query)

    w, h = 150, 110

    #Showing Images

    imageContainer = Frame(chat\_frame, bg='#EAEAEA')

    imageContainer.pack(anchor='w')

    #loading images

    img0 = ImageTk.PhotoImage(Image.open('Downloads/0.jpg').resize((w,h), Image.ANTIALIAS))

    img1 = ImageTk.PhotoImage(Image.open('Downloads/1.jpg').resize((w,h), Image.ANTIALIAS))

    img2 = ImageTk.PhotoImage(Image.open('Downloads/2.jpg').resize((w,h), Image.ANTIALIAS))

    img3 = ImageTk.PhotoImage(Image.open('Downloads/3.jpg').resize((w,h), Image.ANTIALIAS))

    #Displaying

    Label(imageContainer, image=img0, bg='#EAEAEA').grid(row=0, column=0)

    Label(imageContainer, image=img1, bg='#EAEAEA').grid(row=0, column=1)

    Label(imageContainer, image=img2, bg='#EAEAEA').grid(row=1, column=0)

    Label(imageContainer, image=img3, bg='#EAEAEA').grid(row=1, column=1)

############################# WAEM - WhatsApp Email ##################################

def sendWAEM():

    global rec\_phoneno, rec\_email

    data = WAEMEntry.get()

    rec\_email, rec\_phoneno = data, data

    WAEMEntry.delete(0, END)

    appControl.Win\_Opt('close')

def send(e):

    sendWAEM()

def WAEMPOPUP(Service='None', rec='Reciever'):

    global WAEMEntry

    PopUProot = Tk()

    PopUProot.title(f'{Service} Service')

    PopUProot.configure(bg='white')

    if Service=="WhatsApp": PopUProot.iconbitmap("extrafiles/images/whatsapp.ico")

    else: PopUProot.iconbitmap("extrafiles/images/email.ico")

    w\_width, w\_height = 410, 200

    s\_width, s\_height = PopUProot.winfo\_screenwidth(), PopUProot.winfo\_screenheight()

    x, y = (s\_width/2)-(w\_width/2), (s\_height/2)-(w\_height/2)

    PopUProot.geometry('%dx%d+%d+%d' % (w\_width,w\_height,x,y-30)) #center location of the screen

    Label(PopUProot, text=f'Reciever {rec}', font=('Arial', 16), bg='white').pack(pady=(20, 10))

    WAEMEntry = Entry(PopUProot, bd=10, relief=FLAT, font=('Arial', 12), justify='center', bg='#DCDCDC', width=30)

    WAEMEntry.pack()

    WAEMEntry.focus()

    SendBtn = Button(PopUProot, text='Send', font=('Arial', 12), relief=FLAT, bg='#14A769', fg='white', command=sendWAEM)

    SendBtn.pack(pady=20, ipadx=10)

    PopUProot.bind('<Return>', send)

    PopUProot.mainloop()

######################## CHANGING CHAT BACKGROUND COLOR #########################

def getChatColor():

    global chatBgColor

    myColor = colorchooser.askcolor()

    if myColor[1] is None: return

    chatBgColor = myColor[1]

    colorbar['bg'] = chatBgColor

    chat\_frame['bg'] = chatBgColor

    root1['bg'] = chatBgColor

    ChangeSettings(True)

chatMode = 1

def changeChatMode():

    global chatMode

    if chatMode==1:

        appControl.volumeControl('mute')

        VoiceModeFrame.pack\_forget()

        TextModeFrame.pack(fill=BOTH)

        UserField.focus()

        chatMode=0

    else:

        appControl.volumeControl('full')

        TextModeFrame.pack\_forget()

        VoiceModeFrame.pack(fill=BOTH)

        root.focus()

        chatMode=1

#####################################  MAIN GUI ####################################################

#### SPLASH/LOADING SCREEN ####

def progressbar():

    s = ttk.Style()

    s.theme\_use('clam')

    s.configure("white.Horizontal.TProgressbar", foreground='white', background='white')

    progress\_bar = ttk.Progressbar(splash\_root,style="white.Horizontal.TProgressbar", orient="horizontal",mode="determinate", length=303)

    progress\_bar.pack()

    splash\_root.update()

    progress\_bar['value'] = 0

    splash\_root.update()

    while progress\_bar['value'] < 100:

        progress\_bar['value'] += 5

        # splash\_percentage\_label['text'] = str(progress\_bar['value']) + ' %'

        splash\_root.update()

        sleep(0.1)

def destroySplash():

    splash\_root.destroy()

if \_\_name\_\_ == '\_\_main\_\_':

    splash\_root = Tk()

    splash\_root.configure(bg='#3895d3')

    splash\_root.overrideredirect(True)

    splash\_label = Label(splash\_root, text="Processing...", font=('montserrat',15),bg='#3895d3',fg='white')

    splash\_label.pack(pady=40)

    # splash\_percentage\_label = Label(splash\_root, text="0 %", font=('montserrat',15),bg='#3895d3',fg='white')

    # splash\_percentage\_label.pack(pady=(0,10))

    w\_width, w\_height = 400, 200

    s\_width, s\_height = splash\_root.winfo\_screenwidth(), splash\_root.winfo\_screenheight()

    x, y = (s\_width/2)-(w\_width/2), (s\_height/2)-(w\_height/2)

    splash\_root.geometry('%dx%d+%d+%d' % (w\_width,w\_height,x,y-30))

    progressbar()

    splash\_root.after(10, destroySplash)

    splash\_root.mainloop()

    root = Tk()

    root.title('F.R.I.D.A.Y')

    w\_width, w\_height = 400, 650

    s\_width, s\_height = root.winfo\_screenwidth(), root.winfo\_screenheight()

    x, y = (s\_width/2)-(w\_width/2), (s\_height/2)-(w\_height/2)

    root.geometry('%dx%d+%d+%d' % (w\_width,w\_height,x,y-30)) #center location of the screen

    root.configure(bg=background)

    # root.resizable(width=False, height=False)

    root.pack\_propagate(0)

    root1 = Frame(root, bg=chatBgColor)

    root2 = Frame(root, bg=background)

    root3 = Frame(root, bg=background)

    for f in (root1, root2, root3):

        f.grid(row=0, column=0, sticky='news')

    ################################

    ########  CHAT SCREEN  #########

    ################################

    #Chat Frame

    chat\_frame = Frame(root1, width=380,height=551,bg=chatBgColor)

    chat\_frame.pack(padx=10)

    chat\_frame.pack\_propagate(0)

    bottomFrame1 = Frame(root1, bg='#dfdfdf', height=100)

    bottomFrame1.pack(fill=X, side=BOTTOM)

    VoiceModeFrame = Frame(bottomFrame1, bg='#dfdfdf')

    VoiceModeFrame.pack(fill=BOTH)

    TextModeFrame = Frame(bottomFrame1, bg='#dfdfdf')

    TextModeFrame.pack(fill=BOTH)

    # VoiceModeFrame.pack\_forget()

    TextModeFrame.pack\_forget()

    cblLightImg = PhotoImage(file='extrafiles/images/centralButton.png')

    cblDarkImg = PhotoImage(file='extrafiles/images/centralButton1.png')

    if KCS\_IMG==1: cblimage=cblDarkImg

    else: cblimage=cblLightImg

    cbl = Label(VoiceModeFrame, fg='white', image=cblimage, bg='#dfdfdf')

    cbl.pack(pady=17)

    AITaskStatusLbl = Label(VoiceModeFrame, text='    Offline', fg='white', bg=AITaskStatusLblBG, font=('montserrat', 16))

    AITaskStatusLbl.place(x=140,y=32)

    #Settings Button

    sphLight = PhotoImage(file = "extrafiles/images/setting.png")

    sphLight = sphLight.subsample(2,2)

    sphDark = PhotoImage(file = "extrafiles/images/setting1.png")

    sphDark = sphDark.subsample(2,2)

    if KCS\_IMG==1: sphimage=sphDark

    else: sphimage=sphLight

    settingBtn = Button(VoiceModeFrame,image=sphimage,height=30,width=30, bg='#dfdfdf',borderwidth=0,activebackground="#dfdfdf",command=lambda: raise\_frame(root2))

    settingBtn.place(relx=1.0, y=30,x=-20, anchor="ne")

    #Keyboard Button

    kbphLight = PhotoImage(file = "extrafiles/images/keyboard.png")

    kbphLight = kbphLight.subsample(2,2)

    kbphDark = PhotoImage(file = "extrafiles/images/keyboard1.png")

    kbphDark = kbphDark.subsample(2,2)

    if KCS\_IMG==1: kbphimage=kbphDark

    else: kbphimage=kbphLight

    kbBtn = Button(VoiceModeFrame,image=kbphimage,height=30,width=30, bg='#dfdfdf',borderwidth=0,activebackground="#dfdfdf", command=changeChatMode)

    kbBtn.place(x=25, y=30)

    #Mic

    micImg = PhotoImage(file = "extrafiles/images/mic.png")

    micImg = micImg.subsample(2,2)

    micBtn = Button(TextModeFrame,image=micImg,height=30,width=30, bg='#dfdfdf',borderwidth=0,activebackground="#dfdfdf", command=changeChatMode)

    micBtn.place(relx=1.0, y=30,x=-20, anchor="ne")

    #Text Field

    TextFieldImg = PhotoImage(file='extrafiles/images/textField.png')

    UserFieldLBL = Label(TextModeFrame, fg='white', image=TextFieldImg, bg='#dfdfdf')

    UserFieldLBL.pack(pady=17, side=LEFT, padx=10)

    UserField = Entry(TextModeFrame, fg='white', bg='#203647', font=('Montserrat', 16), bd=6, width=22, relief=FLAT)

    UserField.place(x=20, y=30)

    UserField.insert(0, "Ask me anything...")

    UserField.bind('<Return>', keyboardInput)

    #User and Bot Icon

    userIcon = PhotoImage(file="extrafiles/images/avatars/ChatIcons/a"+str(ownerPhoto)+".png")

    botIcon = PhotoImage(file="extrafiles/images/assistant2.png")

    botIcon = botIcon.subsample(2,2)

    ###########################

    ########  SETTINGS  #######

    ###########################

    settingsLbl = Label(root2, text='Settings', font=('Arial Bold', 15), bg=background, fg=textColor)

    settingsLbl.pack(pady=10)

    separator = ttk.Separator(root2, orient='horizontal')

    separator.pack(fill=X)

    #User Photo

    userProfileImg = Image.open("extrafiles/images/avatars/a"+str(ownerPhoto)+".png")

    userProfileImg = ImageTk.PhotoImage(userProfileImg.resize((120, 120)))

    userPhoto = Button(root2, image=userProfileImg, bg=background, bd=0, relief=FLAT, activebackground=background)

    userPhoto.pack(pady=(20, 5))

    #Username

    userName = Label(root2, text=ownerName, font=('Arial Bold', 15), fg=textColor, bg=background)

    userName.pack()

    #Settings Frame

    settingsFrame = Frame(root2, width=300, height=300, bg=background)

    settingsFrame.pack(pady=20)

    assLbl = Label(settingsFrame, text='Assistant Voice', font=('Arial', 13), fg=textColor, bg=background)

    assLbl.place(x=0, y=20)

    n = StringVar()

    assVoiceOption = ttk.Combobox(settingsFrame, values=('Female', 'Male'), font=('Arial', 13), width=13, textvariable=n)

    assVoiceOption.current(voice\_id)

    assVoiceOption.place(x=150, y=20)

    assVoiceOption.bind('<<ComboboxSelected>>', changeVoice)

    voiceRateLbl = Label(settingsFrame, text='Voice Rate', font=('Arial', 13), fg=textColor, bg=background)

    voiceRateLbl.place(x=0, y=60)

    n2 = StringVar()

    voiceOption = ttk.Combobox(settingsFrame, font=('Arial', 13), width=13, textvariable=n2)

    voiceOption['values'] = ('Very Low', 'Low', 'Normal', 'Fast', 'Very Fast')

    voiceOption.current(ass\_voiceRate//50-2) #100 150 200 250 300

    voiceOption.place(x=150, y=60)

    voiceOption.bind('<<ComboboxSelected>>', changeVoiceRate)

    volumeLbl = Label(settingsFrame, text='Volume', font=('Arial', 13), fg=textColor, bg=background)

    volumeLbl.place(x=0, y=105)

    volumeBar = Scale(settingsFrame, bg=background, fg=textColor, sliderlength=30, length=135, width=16, highlightbackground=background, orient='horizontal', from\_=0, to=100, command=changeVolume)

    volumeBar.set(int(ass\_volume\*100))

    volumeBar.place(x=150, y=85)

    themeLbl = Label(settingsFrame, text='Theme', font=('Arial', 13), fg=textColor, bg=background)

    themeLbl.place(x=0,y=143)

    themeValue = IntVar()

    s = ttk.Style()

    s.configure('Wild.TRadiobutton', font=('Arial Bold', 10), background=background, foreground=textColor, focuscolor=s.configure(".")["background"])

    darkBtn = ttk.Radiobutton(settingsFrame, text='Dark', value=1, variable=themeValue, style='Wild.TRadiobutton', command=changeTheme, takefocus=False)

    darkBtn.place(x=150,y=145)

    lightBtn = ttk.Radiobutton(settingsFrame, text='Light', value=2, variable=themeValue, style='Wild.TRadiobutton', command=changeTheme, takefocus=False)

    lightBtn.place(x=230,y=145)

    themeValue.set(1)

    if KCS\_IMG==0: themeValue.set(2)

    chooseChatLbl = Label(settingsFrame, text='Chat Background', font=('Arial', 13), fg=textColor, bg=background)

    chooseChatLbl.place(x=0,y=180)

    cimg = PhotoImage(file = "extrafiles/images/colorchooser.png")

    cimg = cimg.subsample(3,3)

    colorbar = Label(settingsFrame, bd=3, width=18, height=1, bg=chatBgColor)

    colorbar.place(x=150, y=180)

    if KCS\_IMG==0: colorbar['bg'] = '#E8EBEF'

    Button(settingsFrame, image=cimg, relief=FLAT, command=getChatColor).place(x=261, y=180)

    backBtn = Button(settingsFrame, text='   Back   ', bd=0, font=('Arial 12'), fg='white', bg='#14A769', relief=FLAT, command=lambda:raise\_frame(root1))

    clearFaceBtn = Button(settingsFrame, text='   Close the ChatBot   ', bd=0, font=('Arial 12'), fg='white', bg='#14A769', relief=FLAT, command=deleteUserData)

    backBtn.place(x=5, y=250)

    clearFaceBtn.place(x=120, y=250)

    try:

        # pass

        Thread(target=voiceMedium).start()

    except:

        pass

    try:

        # pass

        Thread(target=webScrapping.dataUpdate).start()

    except Exception as e:

        print('System is Offline...')

    root.iconbitmap('extrafiles/images/assistant2.ico')

    raise\_frame(root1)

    root.mainloop()

**Math\_function.py**

import math

def basicOperations(text):

    if 'root' in text:

        temp = text.rfind(' ')

        num = int(text[temp+1:])

        return round(math.sqrt(num),2)

    text = text.replace('plus', '+')

    text = text.replace('minus', '-')

    text = text.replace('x', '\*')

    text = text.replace('multiplied by', '\*')

    text = text.replace('multiply', '\*')

    text = text.replace('divided by', '/')

    text = text.replace('to the power', '\*\*')

    text = text.replace('power', '\*\*')

    result = eval(text)

    return round(result,2)

def bitwiseOperations(text):

    if 'right shift' in text:

        temp = text.rfind(' ')

        num = int(text[temp+1:])

        return num>>1

    elif 'left shift' in text:

        temp = text.rfind(' ')

        num = int(text[temp+1:])

        return num<<1

    text = text.replace('and', '&')

    text = text.replace('or', '|')

    text = text.replace('not of', '~')

    text = text.replace('not', '~')

    text = text.replace('xor', '^')

    result = eval(text)

    return result

def conversions(text):

    temp = text.rfind(' ')

    num = int(text[temp+1:])

    if 'bin' in text:

        return eval('bin(num)')[2:]

    elif 'hex' in text:

        return eval('hex(num)')[2:]

    elif 'oct' in text:

        return eval('oct(num)')[2:]

def trigonometry(text):

    temp = text.replace('degree','')

    temp = text.rfind(' ')

    deg = int(text[temp+1:])

    rad = (deg \* math.pi) / 180

    if 'sin' in text:

        return round(math.sin(rad),2)

    elif 'cos' in text:

        return round(math.cos(rad),2)

    elif 'tan' in text:

        return round(math.tan(rad),2)

def factorial(n):

    if n==1 or n==0: return 1

    else: return n\*factorial(n-1)

def logFind(text):

    temp = text.rfind(' ')

    num = int(text[temp+1:])

    return round(math.log(num,10),2)

def isHaving(text, lst):

    for word in lst:

        if word in text:

            return True

    return False

def perform(text):

    text = text.replace('math','')

    if "factorial" in text: return str(factorial(int(text[text.rfind(' ')+1:])))

    elif isHaving(text, ['sin','cos','tan']): return str(trigonometry(text))

    elif isHaving(text, ['bin','hex','oct']): return str(conversions(text))

    elif isHaving(text, ['shift','and','or','not']): return str(bitwiseOperations(text))

    elif 'log' in text: return str(logFind(text))

    else: return str(basicOperations(text))

**NormalChat.py**

from difflib import get\_close\_matches

import json

from random import choice

import datetime

class DateTime:

    def currentTime(self):

        time = datetime.datetime.now()

        x = " A.M."

        if time.hour>12: x = " P.M."

        time = str(time)

        time = time[11:16] + x

        return time

    def currentDate(self):

        now = datetime.datetime.now()

        day = now.strftime('%A')

        date = str(now)[8:10]

        month = now.strftime('%B')

        year = str(now.year)

        result = f'{day}, {date} {month}, {year}'

        return result

def wishMe():

    now = datetime.datetime.now()

    hr = now.hour

    if hr<12:

        wish="Good Morning"

    elif hr>=12 and hr<16:

        wish="Good Afternoon"

    else:

        wish="Good Evening"

    return wish

def isContain(text, lst):

    for word in lst:

        if word in text:

            return True

    return False

def chat(text):

    dt = DateTime()

    result = ""

    if isContain(text, ['good']):

        result = wishMe()

    elif isContain(text, ['time']):

        result = "Current Time is: " + dt.currentTime()

    elif isContain(text, ['date','today','day','month']):

        result = dt.currentDate()

    return result

data = json.load(open('extrafiles/NormalChat.json', encoding='utf-8'))

def reply(query):

    if query in data:

        response =  data[query]

    else:

        query = get\_close\_matches(query, data.keys(), n=2, cutoff=0.6)

        if len(query)==0: return "None"

        return choice(data[query[0]])

    return choice(response)

def lang\_translate(text,language):

    from googletrans import Translator, LANGUAGES

    if language in LANGUAGES.values():

        translator = Translator()

        result = translator.translate(text, src='en', dest=language)

        return result

    else:

        return "None"

**Timer.py**

from time import sleep

import re

import playsound

from tkinter import \*

from threading import Thread

def startTimer(query):

    nums = re.findall(r'[0-9]+', query)

    time = 0

    if "minute" in query and "second" in query:

        time = int(nums[0])\*60 + int(nums[1])

    elif "minute" in query:

        time = int(nums[0])\*60

    elif "second" in query:

        time = int(nums[0])

    else: return

    print("Timer Started")

    sleep(time)

    Thread(target=timer).start()

    playsound.playsound("extrafiles/audios/Timer.mp3")

def timer():

    root = Tk()

    root.title("Timer")

    root.iconbitmap("extrafiles/images/timer.ico")

    w\_width, w\_height = 300, 150

    s\_width, s\_height = root.winfo\_screenwidth(), root.winfo\_screenheight()

    x, y = (s\_width/2)-(w\_width/2), (s\_height/2)-(w\_height/2)

    root.geometry('%dx%d+%d+%d' % (w\_width,w\_height,x,y-30))

    root['bg'] = 'white'

    Label(root, text="Time's Up", font=("Arial Bold", 20), bg='white').pack(pady=20)

    Button(root, text="  OK  ", font=("Arial", 15), relief=FLAT, bg='#14A769', fg='white', command=lambda:quit()).pack()

    root.mainloop()

**Todo.py**

from datetime import datetime

import os

file = "userData/toDoList.txt"

def createList():

    f = open(file,"w")

    present = datetime.now()

    dt\_format = present.strftime("Date: " + "%d/%m/%Y"+ " Time: " + "%H:%M:%S" + "\n")

    f.write(dt\_format)

    f.close()

def toDoList(text):

    if os.path.isfile(file) == False:

        createList()

    f = open(file,"r")

    x = f.read(8)

    f.close()

    y = x[6:]

    yesterday = int(y)

    present = datetime.now()

    today = int(present.strftime("%d"))

    if (today-yesterday) >= 1:

        createList()

    f = open(file,"a")

    dt\_format = present.strftime("%H:%M")

    print(dt\_format)

    f.write(f"[{dt\_format}] : {text}\n")

    f.close()

def showtoDoList():

    if os.path.isfile(file)==False:

        return ["It looks like that list is empty"]

    f = open(file, 'r')

    items = []

    for line in f.readlines():

        items.append(line.strip())

    speakList = [f"You have {len(items)-1} items in your list:\n"]

    for i in items[1:]:

        speakList.append(i.capitalize())

    return speakList

**userHandler.py**

import pickle

class UserData:

    def \_\_init\_\_(self):

        self.name = 'None'

        self.gender = 'None'

        self.userphoto = 0

    def extractData(self):

        with open('userData/userData.pck', 'rb') as file:

            details = pickle.load(file)

            self.name, self.gender, self.userphoto = details['name'], details['gender'], details['userphoto']

    def updateData(self, name, gender, userphoto):

        with open('userData/userData.pck', 'wb') as file:

            details = {'name': name, 'gender': gender, 'userphoto': userphoto}

            pickle.dump(details, file)

    def getName(self):

        return self.name

    def getGender(self):

        return self.gender

    def getUserPhoto(self):

        return self.userphoto

def UpdateUserPhoto(avatar):

    u = UserData()

    u.extractData()

    u.updateData(u.getName(), u.getGender(), avatar)

**webScrapping.py**

import wikipedia

import webbrowser

import requests

from bs4 import BeautifulSoup

import threading

import smtplib

import urllib.request

import os

from geopy.geocoders import Nominatim

from geopy.distance import great\_circle

class COVID:

    def \_\_init\_\_(self):

        self.total = 'Not Available'

        self.deaths = 'Not Available'

        self.recovered = 'Not Available'

        self.totalIndia = 'Not Available'

        self.deathsIndia = 'Not Available'

        self.recoveredIndia = 'Not Available'

    def covidUpdate(self):

        URL = 'https://www.worldometers.info/coronavirus/'

        result = requests.get(URL)

        src = result.content

        soup = BeautifulSoup(src, 'html.parser')

        temp = []

        divs = soup.find\_all('div', class\_='maincounter-number')

        for div in divs:

            temp.append(div.text.strip())

        self.total, self.deaths, self.recovered = temp[0], temp[1], temp[2]

    def covidUpdateIndia(self):

        URL = 'https://www.worldometers.info/coronavirus/country/india/'

        result = requests.get(URL)

        src = result.content

        soup = BeautifulSoup(src, 'html.parser')

        temp = []

        divs = soup.find\_all('div', class\_='maincounter-number')

        for div in divs:

            temp.append(div.text.strip())

        self.totalIndia, self.deathsIndia, self.recoveredIndia = temp[0], temp[1], temp[2]

    def totalCases(self,india\_bool):

        if india\_bool: return self.totalIndia

        return self.total

    def totalDeaths(self,india\_bool):

        if india\_bool: return self.deathsIndia

        return self.deaths

    def totalRecovery(self,india\_bool):

        if india\_bool: return self.recoveredIndia

        return self.recovered

    def symptoms(self):

        symt = ['1. Fever',

                '2. Coughing',

                '3. Shortness of breath',

                '4. Trouble breathing',

                '5. Fatigue',

                '6. Chills, sometimes with shaking',

                '7. Body aches',

                '8. Headache',

                '9. Sore throat',

                '10. Loss of smell or taste',

                '11. Nausea',

                '12. Diarrhea']

        return symt

    def prevention(self):

        prevention = ['1. Clean your hands often. Use soap and water, or an alcohol-based hand rub.',

                        '2. Maintain a safe distance from anyone who is coughing or sneezing.',

                        '3. Wear a mask when physical distancing is not possible.',

                        '4. Don’t touch your eyes, nose or mouth.',

                        '5. Cover your nose and mouth with your bent elbow or a tissue when you cough or sneeze.',

                        '6. Stay home if you feel unwell.',

                        '7. If you have a fever, cough and difficulty breathing, seek medical attention.']

        return prevention

def wikiResult(query):

    query = query.replace('wikipedia','')

    query = query.replace('search','')

    if len(query.split())==0: query = "wikipedia"

    try:

        return wikipedia.summary(query, sentences=2)

    except Exception as e:

        return "Desired Result Not Found"

class WEATHER:

    def \_\_init\_\_(self):

        #Currently in Lucknow, its 26 with Haze

        self.tempValue = ''

        self.city = ''

        self.currCondition = ''

        self.speakResult = ''

    def updateWeather(self):

        res = requests.get("https://ipinfo.io/")

        data = res.json()

        # URL = 'https://weather.com/en-IN/weather/today/l/'+data['loc']

        URL = 'https://weather.com/en-IN/weather/today/'

        result = requests.get(URL)

        src = result.content

        soup = BeautifulSoup(src, 'html.parser')

        city = ""

        for h in soup.find\_all('h1'):

            cty = h.text

            cty = cty.replace('Weather','')

            self.city = cty[:cty.find(',')]

            break

        spans = soup.find\_all('span')

        for span in spans:

            try:

                if span['data-testid']=="TemperatureValue":

                    self.tempValue = span.text[:-1]

                    break

            except Exception as e:

                pass

        divs = soup.find\_all('div', class\_='CurrentConditions--phraseValue--2xXSr')

        for div in divs:

            self.currCondition = div.text

            break

    def weather(self):

        from datetime import datetime

        today = datetime.today().strftime('%A')

        self.speakResult = "Currently in " + self.city + ", its " + self.tempValue + " degree, with a " + self.currCondition

        return [self.tempValue, self.currCondition, today, self.city, self.speakResult]

c = COVID()

w = WEATHER()

def dataUpdate():

    c.covidUpdate()

    c.covidUpdateIndia()

    w.updateWeather()

##### WEATHER #####

def weather():

    return w.weather()

### COVID ###

def covid(query):

    if "india" in query: india\_bool = True

    else: india\_bool = False

    if "statistic" in query or 'report' in query:

        return ["Here are the statistics...", ["Total cases: " + c.totalCases(india\_bool), "Total Recovery: " + c.totalRecovery(india\_bool), "Total Deaths: " + c.totalDeaths(india\_bool)]]

    elif "symptom" in query:

        return ["Here are the Symptoms...", c.symptoms()]

    elif "prevent" in query or "measure" in query or "precaution" in query:

        return ["Here are the some of preventions from COVID-19:", c.prevention()]

    elif "recov" in query:

        return "Total Recovery is: " + c.totalRecovery(india\_bool)

    elif "death" in query:

        return "Total Deaths are: " + c.totalDeaths(india\_bool)

    else:

        return "Total Cases are: " + c.totalCases(india\_bool)

def latestNews(news=5):

    URL = 'https://indianexpress.com/latest-news/'

    result = requests.get(URL)

    src = result.content

    soup = BeautifulSoup(src, 'html.parser')

    headlineLinks = []

    headlines = []

    divs = soup.find\_all('div', {'class':'title'})

    count=0

    for div in divs:

        count += 1

        if count>news:

            break

        a\_tag = div.find('a')

        headlineLinks.append(a\_tag.attrs['href'])

        headlines.append(a\_tag.text)

    return headlines,headlineLinks

def maps(text):

    text = text.replace('maps', '')

    text = text.replace('map', '')

    text = text.replace('google', '')

    openWebsite('https://www.google.com/maps/place/'+text)

def giveDirections(startingPoint, destinationPoint):

    geolocator = Nominatim(user\_agent='assistant')

    if 'current' in startingPoint:

        res = requests.get("https://ipinfo.io/")

        data = res.json()

        startinglocation = geolocator.reverse(data['loc'])

    else:

        startinglocation = geolocator.geocode(startingPoint)

    destinationlocation = geolocator.geocode(destinationPoint)

    startingPoint = startinglocation.address.replace(' ', '+')

    destinationPoint = destinationlocation.address.replace(' ', '+')

    openWebsite('https://www.google.co.in/maps/dir/'+startingPoint+'/'+destinationPoint+'/')

    startinglocationCoordinate = (startinglocation.latitude, startinglocation.longitude)

    destinationlocationCoordinate = (destinationlocation.latitude, destinationlocation.longitude)

    total\_distance = great\_circle(startinglocationCoordinate, destinationlocationCoordinate).km #.mile

    return str(round(total\_distance, 2)) + 'KM'

def openWebsite(url='https://www.google.com/'):

    webbrowser.open(url)

def jokes():

    URL = 'https://icanhazdadjoke.com/'

    result = requests.get(URL)

    src = result.content

    soup = BeautifulSoup(src, 'html.parser')

    try:

        p = soup.find('p')

        return p.text

    except Exception as e:

        raise e

def youtube(query):

    from youtube\_search import YoutubeSearch

    query = query.replace('play',' ')

    query = query.replace('on youtube',' ')

    query = query.replace('youtube',' ')

    results = YoutubeSearch(query,max\_results=1).to\_dict()

    webbrowser.open('https://www.youtube.com/watch?v=' + results[0]['id'])

    return "Enjoy Sir..."

def googleSearch(query):

    if 'image' in query:

        query += "&tbm=isch"

    query = query.replace('images','')

    query = query.replace('image','')

    query = query.replace('search','')

    query = query.replace('show','')

    webbrowser.open("https://www.google.com/search?q=" + query)

    return "Here you go..."

def sendWhatsapp(phone\_no='',message=''):

    phone\_no = '+91' + str(phone\_no)

    webbrowser.open('https://web.whatsapp.com/send?phone='+phone\_no+'&text='+message)

    import time

    from pynput.keyboard import Key, Controller

    time.sleep(10)

    k = Controller()

    k.press(Key.enter)

def email(rec\_email=None, text="Hello, It's F.R.I.D.A.Y. here...", sub='F.R.I.D.A.Y.'):

    if '@gmail.com' not in rec\_email: return

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

    s.starttls()

    s.login("sender\_email", "sender\_password")

    message = 'Subject: {}\n\n{}'.format(sub, text)

    s.sendmail("sender\_email", rec\_email, message)

    print("Sent")

    s.quit()

def downloadImage(query, n=4):

    query = query.replace('images','')

    query = query.replace('image','')

    query = query.replace('search','')

    query = query.replace('show','')

    URL = "https://www.google.com/search?tbm=isch&q=" + query

    result = requests.get(URL)

    src = result.content

    soup = BeautifulSoup(src, 'html.parser')

    imgTags = soup.find\_all('img', class\_='t0fcAb')

    if os.path.exists('Downloads')==False:

        os.mkdir('Downloads')

    count=0

    for i in imgTags:

        if count==n: break

        try:

            urllib.request.urlretrieve(i['src'], 'Downloads/' + str(count) + '.jpg')

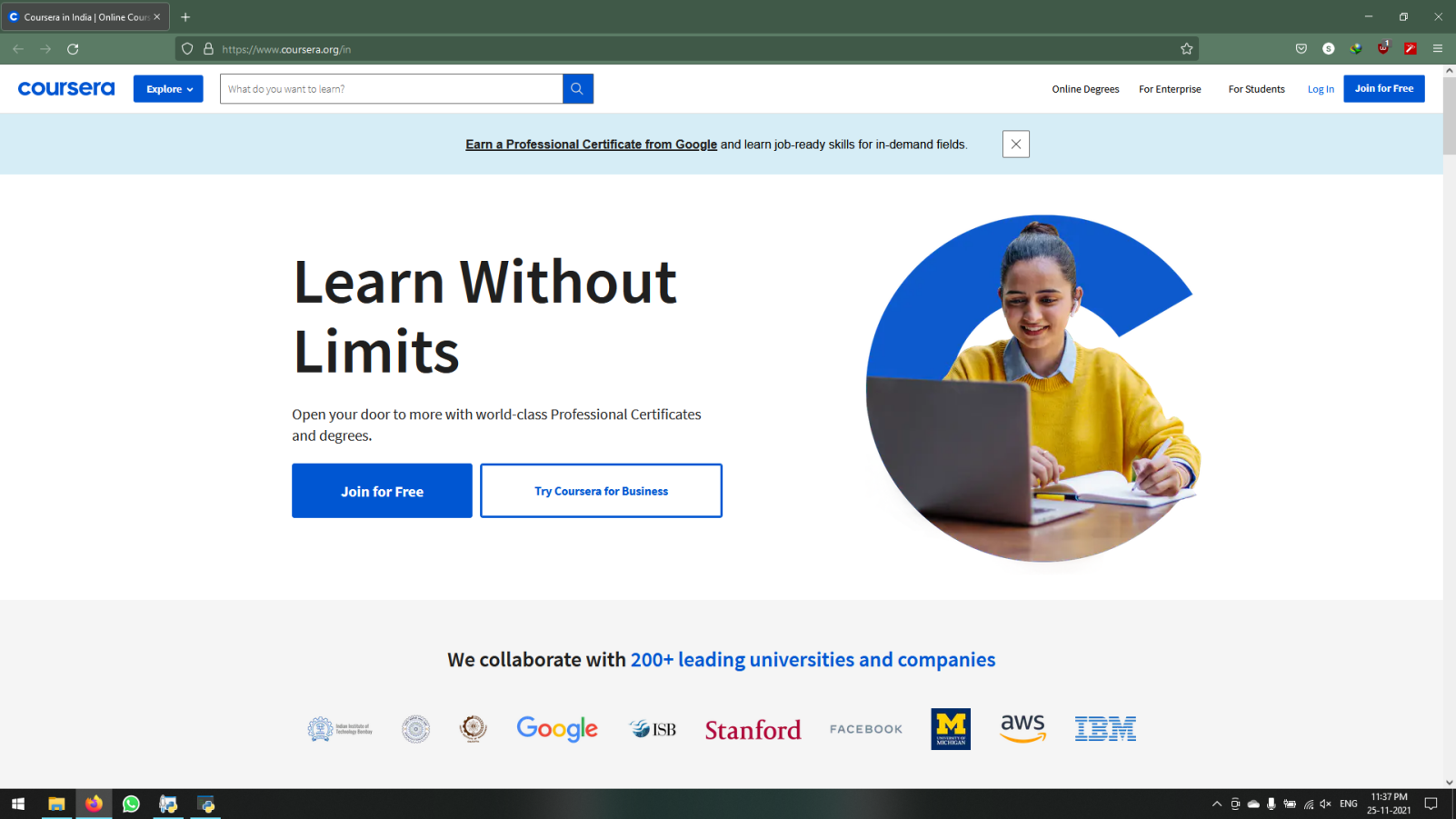
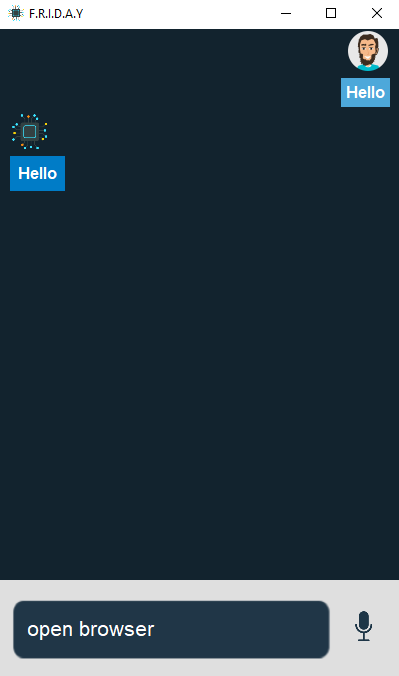
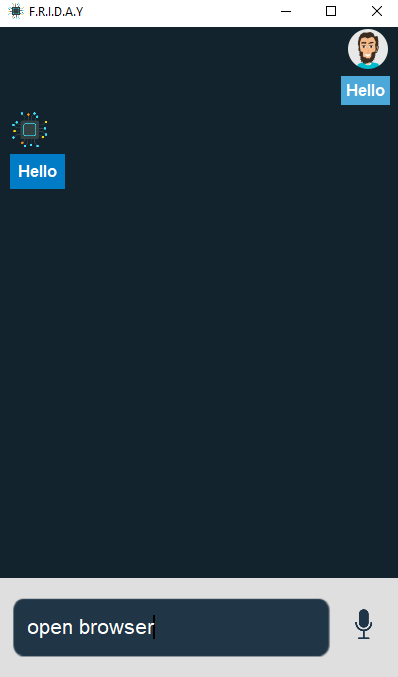
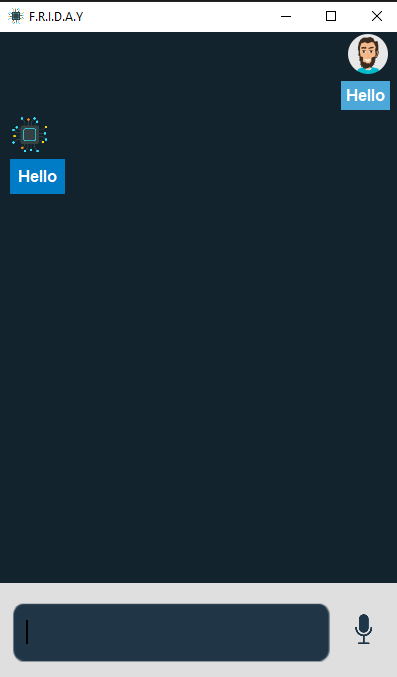
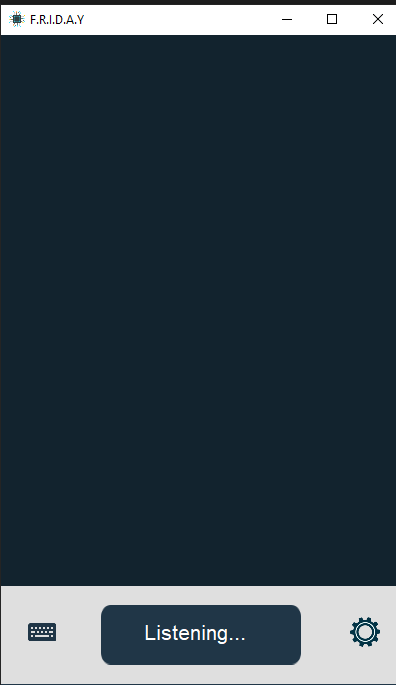
            count+=1

            print('Downloaded', count)

        except Exception as e:

            raise e

**OUTPUT**

****

**Standardization of the coding**

At its core, coding is very similar to any other language in terms of structure and syntax. There

are some laws that must be followed in order for the intricacies of language to be consistent

throughout the world. These guidelines are put into effect so that any new learner of the

language can quickly become acquainted with and comprehend the language in question.

When it comes to coding, there are some concepts to consider, such as readability of code and

maintainability of code. Simply defined, it is when developers follow an identifiable pattern and

set a standard in code that they are considered to be following a pattern. When certain

'regulations' are adhered to, code, like languages, benefits from the metaphorical perspective

of the language.

The earlier you begin, the easier it is to adhere to coding standards. Better to accomplish a

thorough job than to spend time cleaning up after you've finished. Every coder will have his or

her own pattern that he or she will follow. Among the characteristics of his style might be the

conventions he employs to name variables and functions (for example, $userName instead of

$username or $user name), as well as the way he comments on his work. When a pattern and

style are standardized, the effort put in pays off handsomely in the long term.For him, the intrinsic style is second nature, and it can be seen in everything he does and develops. As a

result, the earlier it is standardized, the better.

In the same way that advising developers how to create their code is the same as implementing

a coding standards guideline. Each developer will not code in their own preferred style; instead,

they will adhere to the standards established in this document while writing all of their code.

Thus, the much-desired consistency in coding will be made possible in the future. In order to do

this, the completed source code must have a consistent style, as if it were written by a single

developer in one session. It not only makes the code easier to comprehend, but it also ensures

that any developer who looks at the code will know exactly what to expect throughout the

entire application as a result of this solution. People who maintain code will find it less difficult

to do so if the source code is more readable. The consistency of the style makes it possible for

other developers to step in and assist with maintenance or new development projects.

Making the transition from paper to electronic documentation can be challenging. The majority

of developers would be opposed to the idea and would be defensive about their own personal

coding preferences. In smaller firms, making compromises on the team's preferences and

embracing components of everyone's approach will go a long way toward achieving success.

Developers are likely to embrace the style in stages, much to how a particular accent evolves

over time.The best days of standardization are still ahead of us, to put it another way. An

increasing number of businesses are realizing that applying coding standards can save them

money while also enhancing their efficiency and production. It becomes easier to detect and

correct defects when source code is written in a way that is easier to grasp for a developer to

read. It also provides a more comprehensive picture of how that code integrates into the

overall application.

This improved understanding opens the door to the possibility of greater code reuse, which can

have a significant impact on both cost and development effort. They are no longer regarded as

an afterthought, but rather as a crucial element of the software development process.

Multiplying the number of developers who are familiar with the same code base allows for

speedier implementation while encouraging collaboration on new project requirements.

The advantages of coding standards:

* Integration of the team was simple, which increased the efficiency of the code and made it

easier to maintain.

* Reduce the complexity of the code
* Reduce the cost of development.

It has been said by Guido van Rossum that "code is read far more frequently than it is created."

Creating a piece of code to handle user authentication could take only a few minutes or take an

entire day. The moment you've written something, you're not going to write it again. However,

you will most likely have to read it again. That piece of code may or may not become a part of

the project you're currently working on. You will have to remember what the code does and

why you wrote it every time you return to that file, so readability is important.

You might have a hard time remembering what a piece of code does if you're new to Python

and you've written it a few days, or even a few weeks, ago. If you adhere to PEP 8, you can be

confident that your variables have been appropriately labeled. You'll know when you've added

enough whitespace to your code to make it easier to follow the logical processes in your

programme. You'll have also done a good job of commenting your code. All of these will result

in your code being more understandable and easier to return to in the future. Learning Python

might be a difficult effort for a newbie, but following the requirements of PEP 8 can make the

process lot more enjoyable. Maintaining compliance with PEP 8 is especially vital if you're

seeking for a development position. Writing code that is easy to understand demonstrates

professionalism. It will demonstrate to an employer that you understand how to properly

format your code.

**Testing**

The purpose of including a variety of testing approaches into your development process is to

ensure that your software can operate successfully in a variety of contexts and on a variety of

platforms. Functional testing and non-functional testing are two types of testing that are

commonly used. Functional testing is the process of evaluating an application's functionality in

relation to business requirements. Using use cases provided by the design team or business

analyst, it incorporates all test types needed to ensure that each component of a piece of

software behaves as expected. These testing procedures are normally carried out in a specific

order and comprise the following:

* Testing on a single unit
* Testing the integration of two or more systems
* Testing of the system

Acceptance testing is a type of testing that is performed to ensure that a product meets the

requirements of the customer. Non-functional testing methods include all test kinds that are not focused on the operational features of a piece of software, such as regression testing. These are some examples:

* Testing for overall performance
* Security auditing and testing
* Testing for usability
* Testing for compatibility

**System Security measures**

Cloud-based testing platforms, as well as cyber-attacks, are becoming increasingly popular, and

there is a growing worry and need for the security of data being utilized and kept in software. It

is a type of non-functional software testing method used to verify whether the information and

data included within a system is protected from unauthorized access. By probing the

application for flaws, the goal is to purposely uncover gaps and security risks in the system that

could result in unauthorized access to or loss of information, as opposed to just looking for

them. There are several variations of this testing approach, each of which is intended to verify

the following six fundamental security principles:

* Integrity
* Confidentiality
* Authentication
* Authorization
* Availability
* Non-repudiation

Security Testing Can Be Done in Several Ways

**Scanning for Vulnerabilities**

Automated tools are used to scan for and identify vulnerabilities. It is used to identify known

vulnerabilities in software components, analyse vulnerabilities in order to determine the risk to

the company, and aid in the remedy of vulnerabilities in software components.

**Testing for Penetration (Ethical Hacking)**

It is the practice of simulating realistic cyber assaults against an application, software, system or

network in a controlled environment. Penetration testing is performed in a controlled environment. It can assist in determining how well existing security measures will perform in the event of an actual assault. Penetration testing is particularly useful since it can detect previously undisclosed vulnerabilities, such as zero-day threats and business logic weaknesses.Ethical hackers, who are trustworthy and trained security professionals, have traditionally carried out manual penetration testing on computer systems and networks. An agreed-upon scope is followed by the hacker, who attempts to penetrate a company's systems in a controlled manner while avoiding inflicting any damage. In recent years, automated penetration testing solutions have enabled enterprises to attain similar benefits at a lower cost and with a higher testing frequency than previously possible.

The artificial intelligence-powered penetration testing platform NexPloit is provided by

companies such as NeuraLegion, for example (AI). In addition to automatically scanning various

layers of the IT environment, it also generates reports on vulnerabilities, including zero-day

vulnerabilities and complicated business logic flaws.

**Testing the security of web-based applications**

In order to evaluate whether an online application is vulnerable to attack, web application

security testing is performed. A wide range of automatic and manual approaches are covered in

detail.To acquire information about an online application, find system vulnerabilities or flaws,

study the success of exploiting these flaws or vulnerabilities, and assess the danger of web

application vulnerabilities, penetration testing is carried out.An open-source community

dedicated to detecting and reporting security flaws in web-based applications, the Open Web

Application Security Project (OWASP) was established.

**API Security Testing**

API security testing aids in the identification of vulnerabilities in application programming

interfaces (APIs) and online services, as well as the assistance of developers in the remediation

of those weaknesses. As a means of gaining access to sensitive data, APIs can be exploited by

attackers as an entry point into an organization's internal systems. APIs can be protected from

unauthorized access and exploitation if they are subjected to rigorous and regular testing.

Several threats, such as man in the middle (MiTM) attacks, in which attackers can eavesdrop on

API communications and steal data or credentials, API injections, in which attackers can inject

malicious code into internal systems, and denial of service (DoS), in which attackers flood APIs with fictitious traffic to deny service to legitimate users, are particularly dangerous. APIs must

be verified to ensure that they provide strong authentication of user requests, authorization of

users in accordance with the principle of least privilege, encryption of all communication using

SSL/TLS, and sanitization of user inputs to prevent code injection and tampering, among other

security features.

**Scanning for Configuration**

Security scanning, commonly referred to as configuration scanning, is the technique of finding

misconfigurations in software, networks, and other computing systems by examining their

source code. Systems are often checked against a set of best practices, which are specified by

research groups or compliance standards, during this form of scanning. Tools that perform

automated configuration scanning can find configuration and generate a report that

includes further information on each configuration as well as recommendations on how to

correct them.

**Audits of Information Security**

A security audit is a structured procedure for analyzing and assessing an application or piece of

software in accordance with a recognized standard or set of specifications. The majority of

audits entail reviewing code or designs considering security requirements, identifying security

gaps, and evaluating the security posture of hardware configurations, operating systems, and

organizational practices. It also assesses whether regulations and compliance criteria have been

met.

**Assessment of the Risks**

Risk assessment enables an organization to identify, analyze, and categories the security threats

that it faces in relation to its business-critical assets and assets in general. It is possible to do a

risk assessment to determine which threats are the most significant to an organization's

infrastructure and to priorities the remediation of systems. It can also aid in the long-term

planning and budgeting of security initiatives, as well as the allocation of resources.

**Assessment of the Security Situation**

A security posture assessment combines security scanning, ethical hacking, and risk assessment

to identify not just the risks that an organisation faces, but also the firm's present security

measures and the effectiveness with which they are implemented. It has the capability of

identifying weaknesses in the current security posture and recommending adjustments or

upgrades that will improve the security of assets under protection.

**Documentation**

**Cost Estimation of the Project along with Cost Estimation Model**

The estimated cost of the project will have to be measured in sentimental value as there is no

cost involved in this project. Everything used in this is either open source or very close to the

sources that are incredibly supportive in the community. That does not make the project any

inferior because it uses all the state-of-the-art tech in the algorithm department.

**Pre-Requisite for project Installation**

A working dual core laptop

* SpeechRecognition
* pyttsx3
* playsound
* Pillow
* pyscreenshot
* pynput
* psutil
* wikipedia
* beautifulsoup4
* youtube\_search
* wmi
* geopy
* googletrans
* pyaudio

**Hardware**

Windows 10/Linux OS.

CPU in comparison to Ryzen 3 3rd gen or Intel I3 5 th gen.

Any GPU that is compatible with OpenGL 3.2. (Integrated graphics cards Intel

HD 4000 or above).

4 GB RAM, 10 GB HDD Free Space.

**Software**

An Operating System•

A Web Browser Higher than 2011 Models

Python 3.8 or above

Above Mentioned Libraries.

**Conclusion**

In today’s age, we have completely transferred into the digital age and now a days technology

has become a very integral part of our lives. And since it is so much integrated this project aims

to be one of the best experimental projects for the adults not because of what it offers today

but because of the potential it can offer further in the line. The ability to scan yourself without

acknowledgement and consent can be terrifying but this is supposed to be a mere thought

experiment, OfCourse the addition security measures will be taken when the final

implementation comes out because of the risk that it exposes us to. The AI with the power of

knowing what we don’t even know sometimes. This project detects your mood and plays a song

according to the mood programmed to do a lot more according to your mood, if such an AI

would be to go rogue the consequences would be drastic to say the least that’s why this is

purely a theoretical experiment and to be repeated in extreme care and licensed with even

more care.

**Future Scope**

The future scope and space for this tech is enormous and thanks to the technological

advancement in the field of computer vision and artificial intelligence and deep learning they

are ever-increasing with a speed hither to undreamt of. The one such scenario would be to

have multiple cameras in the room creating a 3D holographic image of the person when they

enter the room and then determining their mood with the help of the algorithms, then proceed

to do the task according to the mood. Its application does not end on the top of being a music

player it can even change basic behavior like the type of movies the user will watch to the

extent to even changing the behavior of the user if integrated with the digital assistant. The scale it can change human behavior is astounding and scary at the same time, for the time

being I can say it is going to change the face of tech if integrated correctly.

**REFRENCES**

1.Player, G. D. (2021). My color of my name”: Composing critical self-celebration with Girls of

Color through a feminist of Color writing pedagogy. Research in the Teaching of English, 55(3),

216-240.

2. Dalmeida, K. M., & Masala, G. L. (2021). Hrv features as viable physiological markers for stress

detection using wearable devices. Sensors, 21(8), 2873.

3. NOMNGA, P., & NGQULU, N. (2021, May). Advancing Artificial Intelligence to Combat Escalating

Cyberspace Human Rights Violations in Africa. In 2021 IST-Africa Conference (IST-Africa) (pp. 1-

9). IEEE.

4. Bahrani, S. M. DEEP LEARNING APPROACH FOR INDIAN CURRENCY CLASSIFICATION.

5.Dalmeida, K. M., & Masala, G. L. (2021). Stress Classification of ECG-Derived HRV Features

Extracted from Wearable Devices.

6. Hjelmås, E., & Low, B. K. (2001). Face detection: A survey. Computer vision and image

understanding, 83(3), 236-274

7. Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. IEEE

Transactions on pattern analysis and machine intelligence, 20(1), 23-38.

8. Yang, S., Luo, P., Loy, C. C., & Tang, X. (2016). Wider face: A face detection benchmark. In

Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5525-5533).

9. Hsu, R. L., Abdel-Mottaleb, M., & Jain, A. K. (2002). Face detection in color images. IEEE

transactions on pattern analysis and machine intelligence, 24(5), 696-706.

10. Viola, P., & Jones, M. (2001, July). Robust real-time face detection. In Proceedings Eighth IEEE

International Conference on Computer Vision (Vol. 3, pp. 747-747). IEEE Computer Society.

11. Yuan, J., Mcdonough, S., You, Q., & Luo, J. (2013, August). Sentribute: image sentiment analysis

from a mid-level perspective. In Proceedings of the Second International Workshop on Issues of

Sentiment Discovery and Opinion Mining (pp. 1-8).

12. Yadav, S. K., Bhushan, M., & Gupta, S. (2015, March). Multimodal sentiment analysis: Sentiment

analysis using audiovisual format. In 2015 2nd International Conference on Computing for

Sustainable Global Development (INDIACom) (pp. 1415-1419). IEEE.

13. Patel, K., Mehta, D., Mistry, C., Gupta, R., Tanwar, S., Kumar, N., & Alazab, M. (2020). Facial

sentiment analysis using AI techniques: state-of-the-art, taxonomies, and challenges. IEEE

Access, 8, 90495-90519.

14. Maynard, D., Dupplaw, D., & Hare, J. (2013). Multimodal sentiment analysis of social media.

15. Kumar, N. S. (2009). SENTIMENT ANALYSIS USING DISCRIMINATIVE MODEL ON TWITTER

DATASET. Education, 2018.

16. Shrivastava, V., Richhariya, V., & Richhariya, V. (2018, December). Puzzling Out Emotions: A

Deep-Learning Approach to Multimodal Sentiment Analysis. In 2018 International Conference

on Advanced Computation and Telecommunication (ICACAT) (pp. 1-6). IEEE.

17. Gajarla, V., & Gupta, A. (2015). Emotion detection and sentiment analysis of images. Georgia

Institute of Technology, 1-4.

18. Dalmeida, K. M., & Masala, G. L. (2021). Hrv features as viable physiological markers for stress

detection using wearable devices. Sensors, 21(8), 2873.

19. Onwujekwe, G. N. (2021). USING DEEP LEARNING-BASED FRAMEWORK FOR CHILD SPEECH

EMOTION RECOGNITION.