ARTIFICIAL INTELLIGENCE

PROJECT

BY-ARNAV BHATIA 02114811622

TANMAY BANSAL 02214811622

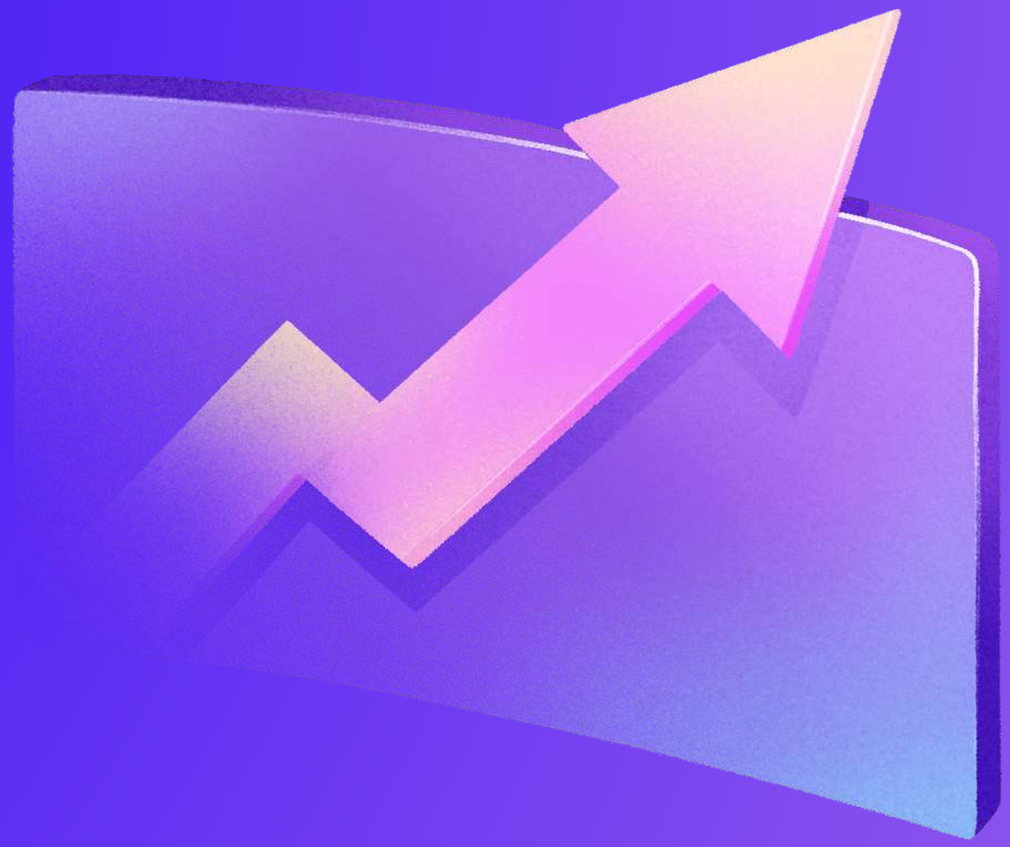
AIML-2ND YEAR

SUBMITED TO – MS GUNJAN CHUGH

TABLE OF CONTENTS

INTRODUCTION

In the contemporary landscape of digital technology, the integration of intelligent and user-centric systems has become paramount to enhance the efficiency and ease of human-computer interactions. Desktop assistants, also known as virtual assistants, represent a significant stride in this direction by providing users with a multifunctional and personalized tool right at their fingertips. This project revolves around the development of a sophisticated desktop assistant using the Python programming language, aiming to amalgamate convenience, functionality, and adaptability in a single, cohesive application.



PROBLEM STATEMENT

The contemporary digital landscape is inundated with a myriad of tasks and information, leading to a growing need for an intelligent and context-aware solution to assist users in managing their desktop activities. Traditional desktop interfaces often lack the sophistication required to adapt to user preferences, comprehend natural language commands, and perform a wide array of tasks seamlessly. This inadequacy results in a less optimized and user-friendly computing experience, where users are burdened with the intricacies of navigating through their systems, searching for information, and performing routine actions.The core problem lies in the absence of a centralized and intelligent system that can understand and respond to user commands in a personalized manner. Existing desktop interfaces often require users to manually execute tasks, search for information, and manage applications, leading to a time-consuming and less intuitive interaction model.

KEY CHALLENGES WITH CURRENT DESKTOP

01

Limited Personalization: Traditional desktop interfaces lack the ability to adapt to individual user preferences,1.resulting in a one-size-fits-all approach that may not align with the diverse needs and workflows of users.

02

Manual Task Execution: Users are often burdened with the manual execution of routine tasks, such as opening applications, searching for files, or setting reminders, leading to a less streamlined and efficient workflow.

03

Integration of Technologies: Integrating technologies such as speech recognition, NLP, and task automation into a cohesive and user-friendly desktop assistant presents a significant technical challenge.

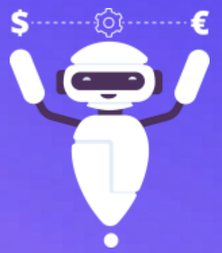
04

Complexity of Commands: Users are constrained by the need to remember specific commands and navigate through complex menu structures, hindering the natural and intuitive interaction with the desktop environment.

PROJECT OBJECTIVES

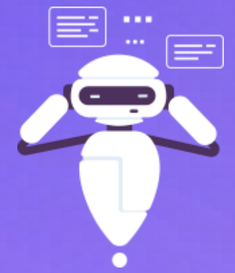
OBJECTIVE 01

Develop a desktop assistant capable of understanding voice and text-based commands



OBJECTIVE 02

Implement functionalities such as opening applications, searching the web, setting reminders, and providing real-time information.



OBJECTIVE 03

Design a modular and extensible architecture to enable easy integration of new features.

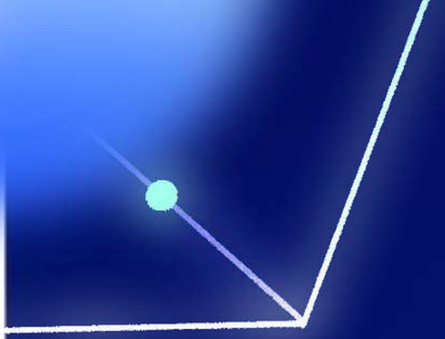
PROJECT SCOPE

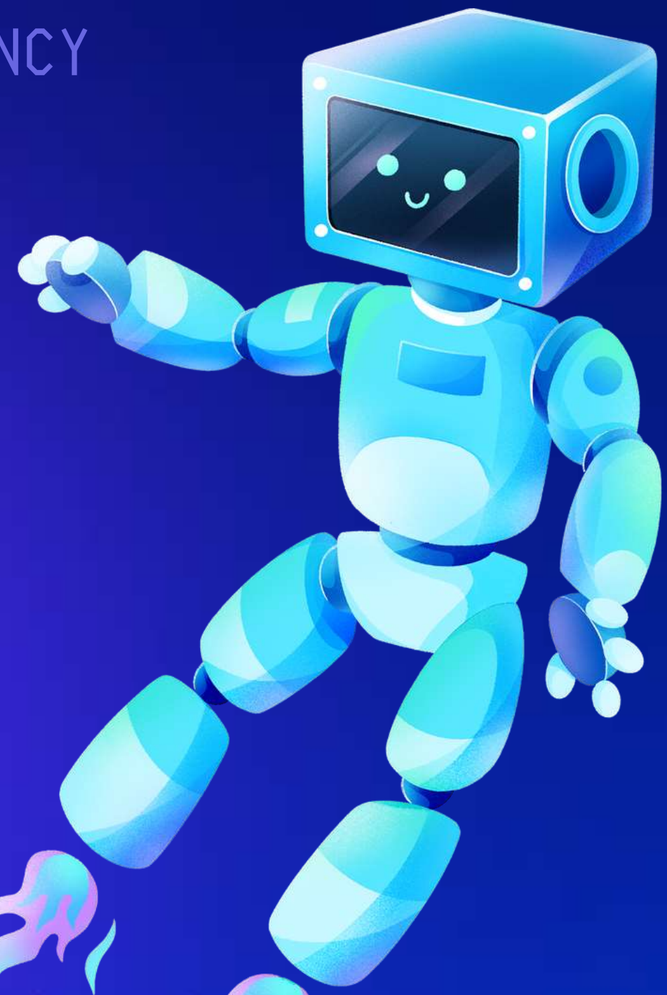
CONTINUOUS LEARNING

INCREASED EFFICIENCY

AUTOMATION

ENHANCED ACCURACY





HARDWARE AND SOFTWARE USED

Hardware:

- Desktop or Laptop computer running Windows, macOS, or Linux.
- Microphone (for voice-based interactions).

Software:

- Python programming language.
- Speech recognition libraries (e.g., SpeechRecognition)..
- Text-to-speech libraries (e.g., pyttsx3).
- Integrated Development Environment (IDE) for Python (e.g., VSCode, PyCharm).
- Version control system (e.g., Git) for collaborative development.
- Use some other basic libraries (eg os, appopener).

.

METHODOLOGY

- Deployment: Make the assistant available for use on various desktop platforms.
- Requirement Analysis: Identify user needs and functionalities
- Implementation: Develop the assistant using Python and relevant libraries.
- Maintenance: Provide ongoing support, updates, and improvements based on user feedback.

Libraries used

- pyttsx3
- speech_recognition
- datetime
- wikipedia
- webbrowser
- os
- AppOpener
- googlesearch
- pywhatkit

```python
import pyttsx3
import speech_recognition as sr
import datetime
import wikipedia
import webbrowser
import os
from AppOpener import open
from googlesearch import search
import pywhatkit as pwt

engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voices', voices[0].id)

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


def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour >= 5 and hour < 12:
        speak("good morning")
    elif hour >= 12 and hour < 16:
        speak("good afternoon")
    elif hour >= 16 and hour <= 23:
        speak("good evening")
    else:
        speak("good night")
    speak("hello I am Tanmay your personal A.I. assistant ")

def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("hearing")
        r.pause_threshold = 1
        audio = r.listen(source)
    try:
        print("recognizing your commands....")
        query = r.recognize_google(audio, language='en-in')
        print(f"user said: {query}\n")
    except Exception as e:
        print(e)
        print("Say that again")
        return "None"
    return query

def googsearch(querry):
    for i in search(querry, tld="com", num=10, stop=10):
        print(i)

```

```python

def ytsearch(querry):
    speak("here is your video")
    pwt.playonyt(querry)


def opengoogle(querry):
    speak("here are your search results")
    pwt.search(querry)


wishMe()
while True:
    query = takeCommand().lower()
    if 'wikipedia' in query:
        speak('searcching Wiki')
        query = query.replace("wikipedia", "")
        results = wikipedia.summary(query, sentences=2)
        speak("Wikipedia says")
        speak(results)
        print(results)
    elif 'stop' in query:
        speak("thank you for using tanmay")
        break
    elif 'open youtube' in query:
        webbrowser.open('http://www.youtube.com')
    elif 'open google' in query:
        webbrowser.open('http://www.google.com')
    elif 'result' in query:
        googsearch(query)
    elif 'search' in query:
        opengoogle(query)
    elif 'play' in query:
        ytsearch(query)
    elif 'open twitter ' in query:
        webbrowser.open('http://www.twitter.com')
    elif 'open map' in query:
        webbrowser.open('https://www.google.com/maps')
    elif 'open gmail' in query:
        webbrowser.open('https://mail.google.com/mail/')
    elif 'open github' in query:
        webbrowser.open('https://github.com/')
    elif 'open stackoverflow' in query:
        webbrowser.open('https://stackoverflow.com/')
    elif 'music' in query:
        music_dir = 'D:\\arnav all\\pai porject\\music'
        songs = os.listdir(music_dir)
        print(songs)
        os.startfile(os.path.join(music_dir, songs[0]))

```

```python
elif 'time' in query:
        strt = datetime.datetime.now().strftime('%H:%M:%S')
        speak(f"the time is {strt}")
    elif 'open code' in query:
        codePath = '"C:\\Users\\91995\\AppData\\Local\\Programs\\Microsoft VS Code\Code.exe"'
        os.startfile(codePath)
    elif 'gta' in query:
        gtapath = 'com.epicgames.launcher://apps//0584d2013f0149a791e7b9bad0eec102:6e563a2c0f5f46e3b4e88b5f4ed50cca:9d2d0eb64d5c44529cece33fe2a46482?action=launch&silent=true'
        os.startfile(gtapath)
    elif 'open canva' in query:
        canvapath = 'C:\\Users\\91995\\AppData\\Local\\Programs\\Canva\\Canva.exe'
        os.startfile(canvapath)
    elif 'open desk' in query:
        daskpath = "D:\\Samsung DeX\\SamsungDeX.exe"
        os.startfile(daskpath)
    elif 'epic game' in query:
        epicpath = "C:\\Program Files (x86)\\Epic Games\\Launcher\\Portal\\Binaries\\Win32\\EpicGamesLauncher.exe"
        os.startfile(epicpath)
    elif 'whatsapp' in query:
        open("whatsapp")
    elif 'game' in query:
        open("forza motorsports 6 apex", match_closest=True)
    elif 'app store' in query:
        open("Microsoft Store", match_closest=True)
    elif 'edge' in query:
        open("Microsoft Edge", match_closest=True)
    elif 'terminal' in query:
        open("Microsoft Terminal", match_closest=True)
    elif 'word' in query:
        open("Microsoft Word", match_closest=True)
    elif 'powepoint' in query:
        open("Microsoft Powepoint", match_closest=True)
    elif 'excel' in query:
        open("Microsoft Excel", match_closest=True)
    elif 'calendar' in query:
        open("Microsoft Calendar", match_closest=True)
    elif 'shutdown' in query:
        shutdown = input(
            "Do you wish to shutdown your computer ? (yes / no): ")
        if shutdown == 'no':
            exit()
        else:
            os.system("shutdown /s /t 1")
```

THANK YOU!

ARNAV BHATIA -02114811622
TANMAY BANSAL- 02214811622