

PROJECT REPORT

DAA Lab.

LEO – Personal Assistant

## ****Submitted To:****

* **Instructor’s Name:** Meghna G Raj Ma’am
* **Position:** ASSISTANT PROFESSOR
* **Department:** SCHOOL OF COMPUTER SCIENCE

## ****Submitted By:****

* **Name : Mohammad Aas Khan**
* **Roll Number:** 22052646
* **Section:** CSE-41

## Overview :-

# Leo is a Python-based personal voice assistant designed to perform a variety of everyday tasks via voice commands. Named "Leo" to convey a helpful and intelligent persona, this assistant integrates essential functions like web searching, system management, entertainment control, and productivity features. Leo uses a modular design and supports various libraries to execute tasks efficiently and provide voice feedback to the user.

# Leo includes functionalities like Google, YouTube, and Wikipedia search, app management, temperature and time checking, alarm setting, and more. The assistant is enhanced with voice output at every step using libraries like pyttsx3, SpeechRecognition, and PyAudio.

## Requirements :-

# Operating System: Windows or Linux

# Python Version: 3.6 or later

# Memory: 1 GB (recommended for smooth performance)

# Storage: 100 MB

## Language and Framework Details :-

# Python is the primary language for Leo, chosen for its simplicity and flexibility in handling automation, voice processing, and web requests.

# Various Libraries such as pyttsx3 for text-to-speech, SpeechRecognition for capturing voice commands, and requests for making web-based requests. Each of these components plays a key role in the assistant’s workflow and modular structure.

## Libraries and Modules Used :-

# The assistant uses a range of libraries to handle voice recognition, text-to-speech, web automation, system control, and more. Each library and module serves a specific purpose:

# Core Libraries:

# pyttsx3: For converting text to speech, which is crucial for providing vocal feedback to the user.

# speech\_recognition: To recognize and interpret spoken commands from the user.

# requests: To send HTTP requests, primarily used to fetch temperature data.

# bs4 (BeautifulSoup): For parsing HTML responses from web searches (temperature checks).

# datetime: To retrieve and format the current date and time.

# os: Used for system-level commands, like shutting down or restarting the computer.

# pyautogui: Allows automation of keyboard actions for controlling YouTube playback and adjusting volume.

# pycaw.pycaw: A Python Core Audio Windows library for directly managing audio interfaces and controlling system volume.

# wikipedia: Used to fetch summaries for quick information.

# webbrowser: Opens URLs in the default web browser.

# pywhatkit: Simplifies YouTube searches and playback.

# time: Used for time-based functions and delays.

## Directory Structure :-

## Leo\_Personal\_AI

|  |  |  |
| --- | --- | --- |
| 1 | main.py | Main script to initialize and run Leo |
| 2 | core\_utils.py | Combined configuration, speak, listen, and hotword detection |
| 3 | conversation.py | Conversation |
| 4 | greet\_user.py | Greeting |
| 5 | web\_search.py | Web search functions (Google, Wikipedia, YouTube) |
| 6 | time.py | Time check function |
| 7 | temperature.py | Temperature check |
| 8 | app\_management.py | Open/close apps on PC |
| 9 | youtube\_control.py | YouTube playback control (play, pause, etc.) |
| 10 | Volume.py | Volume UP and DOWN function |
| 11 | memory.py | Memory functions |
| 12 | system\_control.py | System shutdown, restart, etc. |
| 13 | internet\_speed.py | Internet speed check |
| 14 | game.py | Simple Rock-Paper-Scissors game |

# The project follows a modular design pattern, with separate Python files for each core function. This improves code readability and organization.

## Key Functionalities

## 

# Greeting User:

# Uses greet\_user.py to welcome the user based on the time of day.

# Invokes pyttsx3 to speak the greeting.

# 

# Web Search:

# Google and YouTube Searches: Managed by google\_search and youtube\_search in web\_search.py.

# Wikipedia Search: Uses wikipedia to retrieve summaries.

# Implementation: Uses webbrowser and pywhatkit for YouTube playback.

# 

# Time and Temperature Check:

# Time: Retrieves system time using datetime and formats it for vocal output.

# Temperature: Fetches current temperature using requests and BeautifulSoup.

# 

# Application Management:

# Open and Close Apps: Runs system commands using os to open or close applications.

# 

# YouTube Control:

# Controls playback with keyboard shortcuts managed by pyautogui (play, pause, forward, backward).

# Volume adjustment is achieved by setting specific percentages through pyautogui.

# 

# Memory Management:

# Uses memory.py to store and retrieve simple key-value memory data.

# Uses dictionaries in Python for storing and recalling user-defined information.

# System Control:

# Shutdown and Restart: Uses system\_control.py and os for system shutdown/restart.

# 

# Internet Speed Check:

# Uses speedtest library to check and announce internet speed.

# 

# Game (Rock-Paper-Scissors):

# Implements a basic text-based Rock-Paper-Scissors game as a fun interaction.

# 

# Voice Control:

# Uses core\_utils.py for recognizing voice input and giving voice feedback using speech\_recognition and pyttsx3.

# 

## Conclusion

# The Leo Assistant leverages several Python libraries for creating a multi-functional voice assistant capable of web searches, application management, and system control. The modular structure makes it highly maintainable and extendable, allowing new features to be easily integrated.

# Leo provides a simple, voice-activated interface for performing essential tasks, making it a convenient tool for users looking for a basic voice-activated assistant on their PCs.