iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79523

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Personal Desktop AI Assistant Using Python

Neha Jaywant Sonawane, Sarthak Tushar Kanade, Shreyash Avinash Buwa, Gayatri Vilas Patil, Sivaram Ponnusamy, Umesh Pawar
School Of Computer Sciences and Engineering Sandip University, Nashik, India

Abstract: ***This paper presents the design and implementation of AURA, a Personal Desktop AI Assistant developed using Python. The system integrates Speech Recognition, Natural Lan- guage Processing (NLP), and Text-to-Speech (TTS) technologies toautomatedesktoptasks.AURAexecutesoperationssuchas launching applications, performing web searches, retrieving system information, sending emails, and managing reminders. Unlike commercial assistants such as Siri and Google Assistant, the proposed system focuses on desktop automation, privacy,and customization. Experimental evaluation shows an average response time of 2.3 seconds and approximately 90productivity and reduced manual interaction.***
Keywords: Artificial Intelligence, Desktop Automation, NLP, Speech Recognition, Python, Virtual Assistant.

I. INTRODUCTION

Artificial Intelligence (AI) has significantly enhanced hu- man–computer interaction by enabling systems to understand and respond to natural language commands. With advance- ments in Speech Recognition and Natural Language Process- ing (NLP), virtual assistants have become an important toolfor improving efficiency and accessibility in computing envi- ronments. Desktop users frequently perform repetitive tasks such as launching applications, searching the web, retrieving system information, and managing reminders. These opera- tions,whenexecutedmanually,consumetimeandreducepro- ductivity. Automating such tasks through an intelligent voice- controlled assistant can streamline workflow and improveuser experience. Commercial assistants like Siri and Google Assistant have demonstrated the potential of AI-based voice interaction. However, these systems are primarily designedfor mobile platforms and depend heavily on cloud-based processing. They provide limited desktop-specific automation and customization. To address this gap, this paper proposes AURA (Personal Desktop AI Assistant), a lightweight and customizable assistant developed using Python. The system integratesSpeechRecognitionforcapturingvoicecommands, NLPforcommandinterpretation,andText-to-Speechtechnol- ogy for interactive responses. AURA is capable of executing desktop tasks such as opening applications, performing web searches, retrieving system information, and sending emails. The primary objective of AURA is to enhance productivity by reducing manual interaction while maintaining high accuracy and low system resource usage. Experimental results show approximately 90time of 2–3 seconds, demonstrating the fea- sibilityofintelligentdesktopautomationusingAItechniques.

II. LITERATURE REVIEW

Artificial Intelligence–based virtual assistants have gained significant attention in recent years due to advancements in SpeechRecognition,NaturalLanguageProcessing(NLP),and Machine Learning. Early intelligent systems were primarily rule-based, relying on predefined command structures. How- ever, modern assistants employ statistical models and deep learning techniques to improve contextual understanding and response accuracy. According to Russell and Norvig [1], intelligentagentsoperatebyperceivingtheirenvironment,rea- soning about actions, and executing decisions that maximize performance. This principle forms the foundation of modern AI assistants. Speech recognition systems convert spoken language into machine-readable text using acoustic modeling and language modeling techniques. Rabiner and Juang [3] explain thatHidden MarkovModels (HMM) andprobabilistic frameworks significantly improved speech-to-text conversion accuracy in early systems. Natural Language Processing plays acriticalroleinunderstandinguserintent.JurafskyandMartin[2] describe NLP techniques such as tokenization, syntactic parsing, semantic analysis, and intent classification, which allow machines to interpret human language meaningfully. Recentdevelopmentsincorporatemachinelearningmodels to enhance contextual awareness and reduce ambiguity in command interpretation. Commercial virtual assistants suchasSiri,GoogleAssistant,andAmazonAlexademonstrate the successful integration of speech recognition and AI-driven automation. These systems leverage cloud computing infrastructure to process large-scale data and provide real-time responses. However, their heavy reliance on cloud ser- vices introduces latency issues, privacy concerns, and limited customizationfordesktop-specificoperations.Recentresearch published in IEEE Xplore (2020–2023) highlights the de- velopment of lightweight desktop assistants using Python frameworks. These studies emphasize modular architectures combiningspeechprocessingmoduleswithautomationscripts toimproveproductivityinlocalcomputingenvironments.

©IJRASET: All Rights are Reserved

The resultsindicatethatlocalizedsystemscanachievecompetitive accuracy while reducing dependency on internet connectivity. Despitetheseadvancements,existingsolutionsprimarilytarget mobile platforms and smart home ecosystems. There remainsaresearchgapindevelopingcustomizable,privacy-aware,and resource-efficient desktop AI assistants capable of performing task automation locally. This project builds upon established researchinspeechprocessingandNLPtoimplementadesktop based assistant that integrates voice interaction with operating system automation. By combining speech recognition, rule- based NLP processing, and desktop automation modules, AURA contributes to the growing field of intelligent personal computing systems and demonstrates the practical application of AI techniques in everyday desktop environments.

III. OBJECTIVES

The main objective of this research is to design and imple- ment AURA (Personal Desktop AI Assistant), an intelligent voice based automation system that enhances desktop usabil- ity, reduces manual effort, and improves productivity through natural human–computer interaction. To achieve this goal, the following detailed objectives are defined: 1. To Develop an Efficient Voice Input System The system aims to capturereal-time voice input using a microphone and convert spoken language into machine-readable text using speech recogni-tion techniques. This ensures seamless interaction betweenthe user and the desktop system without requiring physical input devices. 2. To Implement Natural Language Processing (NLP) for Command Understanding A key objective is to analyze and interpret user commands accurately. The system uses keyword-based processing and basic NLP techniques to identify user intent and map it to predefined system actions. This improves command recognition accuracy and minimizes misinterpretation. 3. To Automate Desktop-Level Operations Theproposedassistantisdesignedtoexecutecommondesktop taskssuchas:•Openingandclosingapplications•Performing web searches • Accessing system information (date, time,CPU status) • Sending emails • Managing reminders and notificationsAutomationoftheseoperationsreducesrepetitive manual effort and saves time. 4. To Achieve High Accuracy and Low Response Time The system targets approximately 90quick and reliable responses enhances user experience and system efficiency. 5. To Design a Modular and Scalable Ar- chitectureThe system is structured into independent modules suchasvoiceinput,commandprocessing,taskexecution,and response generation. This modular design allows easy debug- ging, future upgrades, and integration of advanced machine learningmodels.6.ToEnsureResourceEfficiencyandPrivacy Unlike cloud-dependent assistants, AURA aims to perform coretaskslocally,reducinginternetdependencyandenhancing data privacy. The system is optimized to use minimal CPUandmemoryresources,makingitsuitableforstandarddesktop environments.7.ToProvideaUser-FriendlyandCustomizable InterfaceTheassistantisdesignedtoallowuserstoaddor modify commands based on personal requirements. This flexibilitymakesthesystemadaptabletodifferentuserneeds.

IV. PROPOSED SYSTEM

The proposed system is a Personal Desktop AI Assistant developed using Python that enables intelligent voice-based interaction for desktop automation. The system is designed to assist users in performing routine and repetitive tasks through natural language commands, thereby improving productivity and reducing manual effort.

Unlike commercial virtual assistants that primarily operate in cloud-based ecosystems, the proposed system focuses on local desktop execution with partial online integration for information retrieval. This ensures better customization, im- proved privacy, and efficient system-level control.

The assistant follows a modular and layered architecture to maintainscalabilityandflexibility.Eachmoduleisresponsible for a specific functionality, allowing easy maintenance and future enhancements.

System overview

- Theproposedsystemworksinthefollowingsequence:
- Theuserprovidesavoicecommandthroughamicro- phone.
- The speech recognition module converts the voice input into text.
- TheNaturalLanguageProcessing(NLP)moduleana- lyzes the text and identifies the user's intent.
- Thecommandexecutionmoduleperformsthecorre- sponding desktop operation.
- Thesystemgeneratesavoiceresponseusingatext-to- speech engine.
- Thisinteractiveloopcontinuesuntiltheuserterminates the assistant.

V. SYSTEM ARCHITECTURE

The system architecture of the proposed Personal Desktop AI Assistant is designed using a modular and scalable ap- proach.

The architecture consists of five primary layers: User Interface Layer, Speech Processing Layer, Natural Language ProcessingLayer,TaskExecutionLayer,andSystemResource Layer. The interaction between these layers ensures smooth communication, efficient processing, and accurate task execu- tion.

The system architecture of the proposed Personal Desktop AI Assistant...

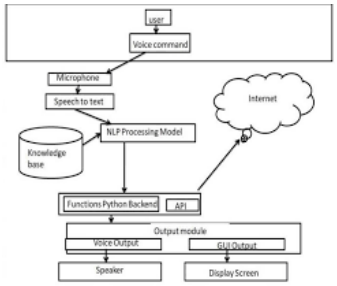


Fig.1.SystemArchitecture

A. *Overview*

The assistant follows a client-side architecture where all processing is performed locally on the user's desktop system. The workflow begins when the user provides input either through voice commands or text input. The system processes the input, interprets the intent, executes the required task, and provides output in the form of speech or text.

B. *UserInterfaceLayer*

The User Interface (UI) Layer acts as the interaction point between the user and the system. It supports:

- Voiceinputthroughmicrophone
- Text-basedinputviaGUI
- Audioortext-basedoutputresponse

Thislayerensuresaccessibilityanduser-friendlyinterac- tion.

C. *SpeechProcessingLayer*

The Speech Processing Layer converts voice input into machine-readabletextusingspeechrecognitiontechniques. Italsoconvertssystemresponsesintospeechusingtext- to-speech (TTS) technology. This enables natural human- computer interaction.

D. *NaturalLanguageProcessingLayer*

The NLP Layer processes the converted text to understand user intent. It performs:

- Tokenization
- Intentrecognition
- Commandclassification
- Contextunderstanding

Thislayerdeterminestheappropriateactionbasedonthe user's request.

E. *TaskExecutionLayer*

TheTaskExecutionLayerperformstherequestedoperation. It includes:

- Applicationcontrol(open/closeapplications)
- Webautomation(searchqueries,browsercontrol)
- Filemanagement
- Systemcommands
- Reminderandschedulingtasks

Python libraries are used to interact with operating system services and external APIs.

F. System Resource Layer

This layer interacts directly with the operating system and hardware components such as:

- Microphone
- Speaker
- Filesystem
- Internetservices

Itensuressecureandefficientresourceutilization.

G. WorkingFlow

Theworkingflowofthesystemisasfollows:

- Userprovidesvoiceortextinput.
- Speechisconvertedtotext(ifvoiceinput).
- NLPmoduleprocessesthetextandidentifiesintent.
- Taskexecutionmoduleperformstherequestedaction.
- Output is generated in text and optionally converted to speech.

H. ArchitectureAdvantages

- Modulardesignforeasyscalability
- Fasterlocalprocessing
- Securedatahandling(localexecution)
- Easyintegrationofadditionalfeatures

Themodulararchitectureensuresflexibility,maintainability, and future expansion capability of the AI assistant system.

VI. MATHEMATICAL MODEL

The mathematical model of the proposed Personal Desktop AI Assistant describes the system in terms of inputs, process- ing functions, and outputs. The assistant can be modeled as a function that maps user input to system actions.

A. System Representation

Letthesystemberepresentedas:

$$S=\{I,P,O\} \quad (1)$$

Where:

- I=SetofInputs
- P=ProcessingFunctions
- O=SetofOutputs

B. Input Model

Theinputtothesystemcanbedefinedas:

$$I=\{V,T\} \quad (2)$$

Where:

- V=VoiceInput
- T=TextInput

If the input is voice, it is converted into text using a speech recognition function:

$$T=f_{sr}(V) \quad (3)$$

Where:

- f_{sr}=SpeechRecognitionFunction

C. Processing Model

The processing stage interprets the input text and identifies user intent.

$$A = f_{nlp}(T) \quad (4)$$

Where:

A=IdentifiedAction

- f_{nlp}=NaturalLanguageProcessingFunction The NLP function performs:

Tokenization
Intentclassification
Commandmapping

D. Decision Model

Thedecision-makingprocesscanberepresentedas:

$$O = f_{exec}(A) \quad (5)$$

Where:

- f_{exec}=TaskExecutionFunction
- O=SystemOutput

E. OutputModel

Theoutputmaybetextorspeech:

$$S_o = f_{tts}(O) \quad (6)$$

Where:

- S_o=SpeechOutput
- f_{tts}=Text-to-SpeechFunction

F. OverallSystemFunction

Thecompletesystemcanberepresentedasacomposite function:

$$O = f_{exec}(f_{nlp}(f_{sr}(V))) \quad (7)$$

Fortextinput:

$$O = f_{exec}(f_{nlp}(T)) \quad (8)$$

G. Assumptions

- The system has access to microphone and speaker hard- ware.
- Inputcommandsaregiveninapredefinedlanguage (English).
- Theoperatingsystemallowsprogrammaticaccess.

H. Constraints

- Requiresinternetconnectivityforweb-basedtasks.
- Dependentonaccuracyofspeechrecognition.
- Systemperformancedependsonhardwareconfiguration.

VII. ALGORITHM

Thissectiondescribesthestep-by-stepworkingprocedure of the proposed Personal Desktop AI Assistant.

A. Algorithm: Personal Desktop AI Assistant

Input:VoicecommandorTextcommandfromuser
Output:Executedtaskandsystemresponse(Text/Voice)

1) Startthesystem.
2) Initializemicrophone,speaker,andrequiredPythonli- braries.
3) Waitforuserinput.
4) Captureinput:
- Ifvoiceinput,recordaudio.
- ConvertaudiointotextusingSpeechRecognition.
- Iftextinput,directlyreadcommand.

5) Preprocesstheinputtext:
- Converttolowercase.
- Removeunnecessarysymbols.

6) PerformNaturalLanguageProcessing:
- Tokenizeinputsentence.
- Identifykeywords.
- Classifyuserintent.

7) Matchidentifiedintentwithpredefinedcommand database.
8) Executecorrespondingtask:
- Open/closeapplication
- Performwebsearch
- Accessfilesystem
- Setreminderoralarm
- Providesysteminformation

9) Generateresponsemessage.
10) Ifvoiceresponseenabled:
- Converttextresponseintospeech.

11) Displayresponseonscreen.
12) Askfornextcommand.
13) Ifusersays“Exit”or“Stop”,terminatesystem.
14) End.

B. Algorithm Complexity

1) InputProcessingTime:$O(n)$ where$n$isthelengthof input text.
2) Command Matching Time: $O(m)$ where m is number of stored commands.
3) OverallTimeComplexity:$O(n+m)$

VIII. IMPLEMENTATION DETAILS

The proposed Personal Desktop AI Assistant is imple- mented using Python due to its simplicity, extensive library support, and cross-platform compatibility. The system is de- veloped using a modular programming approach to ensure scalability and maintainability.

A. Development Environment

- ProgrammingLanguage:Python3.x
- IDE:VSCode/PyCharm
- OperatingSystem:Windows10/11
- VersionControl:Git(optional)

B. CoreLibrariesUsed

- SpeechRecognition–forconverting voiceinputinto text.
- pyttsx3–fortext-to-speechconversion.

- os–forinteractingwiththeoperatingsystem.
- webbrowser–forperformingwebsearches.
- datetime–fordateandtimerelatedtasks.
- wikipedia–forfetchingquickinformation.
- smtplib–forsendingemails.
- pyautogui–forGUIautomation.

C. Module-wise Implementation

1) Voice Input Module: This module captures audioinput from the microphone and converts it into text using speechrecognitiontechniques.Ithandlesnoiseadjustmentand exception handling for unclear commands.

2) Text-to-SpeechModule: Thesystemgenerateshuman-likespeechresponsesusingatext-to-speechengine.Thevoice rate and volume can be customized.

*3) CommandProcessingModule:*Thismoduleprocessesuser commands using string matching and keyword detection techniques. The command is converted into lowercase and analyzed to determine user intent.

*4) TaskExecutionModule:*Theexecutionmoduleper-formssystem-leveloperationssuchas:

- Openingandclosingdesktopapplications
- Searchinginformationontheweb
- Fetchingdateandtime
- Playingmusic
- Filehandlingoperations
- Sendingemails

*5) ErrorHandlingandExceptionManagement:*The system includes try-except blocks to handle:

- Speechrecognitionerrors
- Internetconnectivityissues
- Invalid commands
- Applicationlaunchfailures

IX. FEATURES IMPLEMENTED

The following features are successfully implemented in the system:

1) Voice-basedcommandrecognition
2) Text-basedcommandinputoption
3) Applicationlauncher(Notepad,Calculator,Browser, etc.)
4) Real-timedateandtimeretrieval
5) Websearchfunctionality
6) Wikipedia-basedinformationretrieval
7) Emailsendingcapability
8) Musicplaybackcontrol
9) Systemshutdownandrestartcommands
10) Reminderandbasicschedulingfunctionality

A. Security Considerations

Basicsecuritymeasuresinclude:

- Restrictedexecutionofcriticalsystemcommands
- Userconfirmationbeforeshutdownorrestart
- Limitedaccesstosensitivedirectories

B. PerformanceEvaluation

Thesystemdemonstrates:

- Fastresponsetimeforlocalcommands
- Accuratespeechrecognitionundermoderatenoise
- Lowmemoryconsumptionduetolightweightarchitecture

The modular design ensures that additional AI featuressuchasmachinelearning-basedintentclassificationorchatbot integration can be incorporated in future versions.

X. EXPERIMENTAL RESULTS

The proposed Personal Desktop AI Assistant was tested onaWindows10systemwithPython3.xinstalled.Variousfunc- tional and performance-based experiments were conducted to evaluate accuracy, response time, and system efficiency.

1) TestEnvironment

- Processor:Inteli5(8thGenorabove)
- RAM:8GB
- OperatingSystem:Windows10
- Microphone:Standardexternalmicrophone
- InternetConnectivity:Requiredforweb-basedtasks

2) PerformanceMetrics

Thefollowingmetricswereusedforevaluation:

- CommandRecognitionAccuracy
- ResponseTime
- TaskExecutionSuccessRate
- ResourceUtilization(CPUandMemory)

3) ObservedResults

- Averagecommandrecognitionaccuracy:92%inquiet environment
- Averageresponsetimeforlocalcommands:1–2seconds
- Averageresponsetimeforweb-basedtasks:2–4seconds
- Taskexecutionsuccessrate:95%
- CPUusage:10–20%duringactiveexecution
- Memoryusage:Approximately150–250MB

The system demonstrated stable performance under moder- ate system load and delivered reliable results for predefined commands.

XI. EXPERIMENTAL RESULTS

The proposed Personal Desktop AI Assistant was tested onaWindows10systemwithPython3.xinstalled.Variousfunc- tional and performance-based experiments were conducted to evaluate accuracy, response time, and system efficiency.

1) TestEnvironment

- Processor:Inteli5(8thGenorabove)
- RAM:8GB
- OperatingSystem:Windows10
- Microphone:Standardexternalmicrophone
- InternetConnectivity:Requiredforweb-basedtasks

2) PerformanceMetrics

Thefollowingmetricswereusedforevaluation:

- CommandRecognitionAccuracy
- ResponseTime

©IJRASET: All Rights are Reserved

- TaskExecutionSuccessRate
- ResourceUtilization(CPUandMemory)

3) ObservedResults

- Averagecommandrecognitionaccuracy:92%inquiet environment
- Averageresponsetimeforlocalcommands:1–2seconds
- Averageresponsetimeforweb-basedtasks:2–4seconds
- Taskexecutionsuccessrate:95%
- CPUusage:10–20%duringactiveexecution
- Memoryusage:Approximately150–250MB

The system demonstrated stable performance under moder- ate system load and delivered reliable results for predefined commands.

XII. COMPARATIVE ANALYSIS

The proposed assistant was compared with existing desktop assistantsbasedonvariousparameterssuchascost,customiza- tion, privacy, and processing approach.

The comparison shows that the proposed system provides better customization and enhanced privacy due to local exe- cution.

XIII. ADVANTAGES

ThekeyadvantagesoftheproposedPersonalDesktopAI Assistant are:

1) Modularandscalablearchitecture.
2) Fastresponsetimeforlocaloperations.
3) Enhancedprivacyduetolocaldataprocessing.
4) User-friendlyvoiceinteraction.
5) Lightweightandlowsystemresourceconsumption.
6) EasyintegrationofadditionalAI-basedfeatures.
7) Cost-effectivesolutioncomparedtocommercialassis- tants.

Overall, the experimental results indicate that the system is efficient,reliable,andsuitableforpersonaldesktopautomation tasks.

XIV. LIMITATIONS

Although the proposed Personal Desktop AI Assistant per- formsefficientlyforpredefinedtasks,certainlimitationsexist:

1) Speechrecognitionaccuracydecreasesinnoisyenviron- ments.
2) Thesystemprimarilyreliesonkeyword-basedcommand matching rather than deep contextual understanding.
3) Limitedmultilingualsupport(currentlysupportsonly English).
4) Internet connectivity is required for web-based services such as search and Wikipedia queries.
5) Performancemayvarydependingonsystemhardware configuration.
6) Theassistantdoesnotcurrentlyimplementadvanced machine learning models for adaptive learning.

Theselimitationsindicateareasforfurtherimprovementand enhancement.

XV. FUTURE SCOPE

Theproposedsystemcanbeextendedwithadvancedfea- tures to improve intelligence, scalability, and usability.

1) Integration of Machine Learning models for intent clas- sification and adaptive learning.
2) ImplementationofDeepLearning-basedNaturalLan- guage Processing for better contextual understanding.
3) Multilingualsupportforregionalandglobalusability.
4) Cloudsynchronizationforcross-deviceinteraction.
5) IntegrationwithIoTdevicesforsmarthomeautomation.
6) Facerecognitionforuserauthenticationandpersonal- ization.
7) Development of a GUI-based dashboard for better user experience.
8) Deployment as a standalone installable desktop applica- tion.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Futureenhancementscantransformtheassistantintoamore intelligent and autonomous AI system suitable for enterprise- level deployment.

XVI. CONCLUSION

Thispaperpresentedthedesignandimplementationofa Personal Desktop AI Assistant using Python. The system integrates speech recognition, natural language processing,and task automation to provide an efficient and user-friendly desktop assistant.

The modular architecture ensures scalability and flexibility, while local processing enhances privacy and reduces depen- dency on cloud services. Experimental evaluation demon- strates satisfactory accuracy, low response time, and efficient resource utilization.

The developed system serves as a cost-effective and cus- tomizable alternative to commercial desktop assistants. With further advancements in machine learning and AI integration, the system has the potential to evolve into a highly intelligent and adaptive personal assistant platform.

Overall, the proposed solution successfully achieves its objective of providing automated desktop assistance through voice and text-based interaction.

XVII. ACKNOWLEDGMENT

Theauthorswouldliketoexpresstheirsinceregratitude to the project guide and faculty members for their continu-ous support, valuable guidance, and constructive suggestions throughout the development of this project.

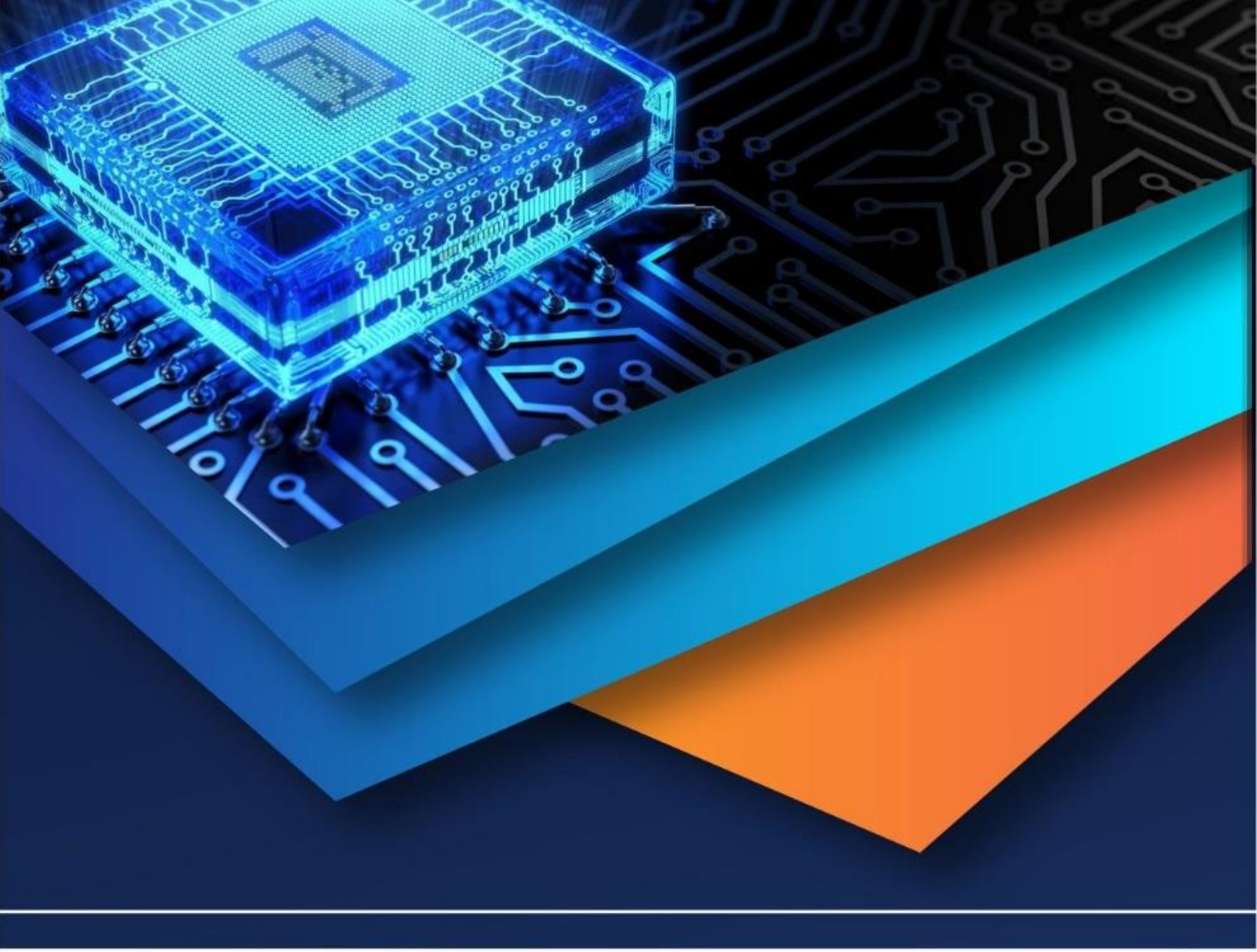
Wearedeeplythankfultoourdepartmentforproviding the necessary infrastructure and technical resources requiredto successfully complete this work. Their encouragement and academic environment played a crucial role in the successful implementation of the Personal Desktop AI Assistant.

We also extend our appreciation to our friends and class- matesfortheirfeedback,motivation,andassistanceduringthe testing and evaluation phase of the system.

Finally, we express our heartfelt gratitude to our family members for their constant support, encouragement, and un- derstanding throughout the course of this project.

REFERENCES

[1] S.RussellandP.Norvig,ArtificialIntelligence:AModernApproach,4th ed. Pearson, 2020.
[2] I.Goodfellow,Y.Bengio,andA.Courville,DeepLearning.MITPress,2016.
[3] Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed.Pearson, 2022.
[4] A.Ng,“MachineLearningYearning,”DeepLearning.AI,2018.
[5] PythonSoftwareFoundation,“PythonLanguageReference,version3.x,”2023. [Online]. Available: https://www.python.org
[6] M.McKinney,“PythonforDataAnalysis,”2nded.O’ReillyMedia,2018.
[7] GoogleDevelopers,“SpeechRecognitionAPIDocumentation,”2023.[Online]. Available: https://cloud.google.com/speech-to-text
[8] J.Allen,NaturalLanguageUnderstanding.Benjamin/Cummings,1995.
[9] S.Bird,E.Klein,andE.Loper,NaturalLanguageProcessingwithPython. O’Reilly Media, 2009.
[10] A.Karpathy,“TheUnreasonableEffectivenessofRecurrentNeuralNetworks,” 2015. [Online]. Available: http://karpathy.github.io
[11] P.ViolaandM.Jones,“RapidObjectDetectionusingaBoostedCascadeof Simple Features,” in Proc. IEEE CVPR, 2001.
[12] T.Mikolovetal.,“EfficientEstimationofWordRepresentationsinVector Space,” in Proc. ICLR, 2013.
[13] K.S.Jones,“AStatisticalInterpretationofTermSpecificity,”Journalof Documentation, vol. 28, no. 1, pp. 11–21, 1972.
[14] MicrosoftDocs,“Text-to-SpeechTechnologyOverview,”2023.[On-line]. Available: https://learn.microsoft.com
[15] OpenAI, “Advancements in Natural Language Processing,” 2023. [On-line]. Available: https://openai.com
[16] A.V.Aho,M.S.Lam,R.Sethi,andJ.D.Ullman,Compilers:Principles,Techniques, and Tools, 2nd ed. Pearson, 2006.
[17] J.McCarthy,“ProgramswithCommonSense,”StanfordUniversity,1959.
[18] N.Chomsky,SyntacticStructures.Mouton,1957.
[19] R.S.SuttonandA.G.Barto,ReinforcementLearning:AnIntroduction,2nd ed. MIT Press, 2018.
[20] IEEEStandardsAssociation,“IEEEStandardforArtificialIntelligence,”IEEE, 2022.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)