



# 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](http://www.ijraset.com)

Call:  08813907089

E-mail ID: [ijraset@gmail.com](mailto:ijraset@gmail.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 Language Processing (NLP), and Text-to-Speech (TTS) technologies to automate desktop tasks. AURA executes operations such as 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 90% productivity and reduced manual interaction.

**Keywords:** Artificial Intelligence, Desktop Automation, NLP, Speech Recognition, Python, Virtual Assistant.

## I. INTRODUCTION

Artificial Intelligence (AI) has significantly enhanced human-computer interaction by enabling systems to understand and respond to natural language commands. With advancements in Speech Recognition and Natural Language Processing (NLP), virtual assistants have become an important tool for improving efficiency and accessibility in computing environments. Desktop users frequently perform repetitive tasks such as launching applications, searching the web, retrieving system information, and managing reminders. These operations, when executed manually, consume time and reduce productivity. Automating such tasks through an intelligent voice-controlled assistant can streamline workflow and improve user experience. Commercial assistants like Siri and Google Assistant have demonstrated the potential of AI-based voice interaction. However, these systems are primarily designed for 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 integrates Speech Recognition for capturing voice commands, NLP for command interpretation, and Text-to-Speech technology 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 an average response time of 2–3 seconds, demonstrating the feasibility of intelligent desktop automation using AI techniques.

## II. LITERATURE REVIEW

Artificial Intelligence-based virtual assistants have gained significant attention in recent years due to advancements in Speech Recognition, Natural Language Processing (NLP), and Machine Learning. Early intelligent systems were primarily rule-based, relying on predefined command structures. However, modern assistants employ statistical models and deep learning techniques to improve contextual understanding and response accuracy. According to Russell and Norvig [1], intelligent agents operate by perceiving their environment, reasoning 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 that Hidden Markov Models (HMM) and probabilistic frameworks significantly improved speech-to-text conversion accuracy in early systems. Natural Language Processing plays a critical role in understanding user intent. Jurafsky and Martin [2] describe NLP techniques such as tokenization, syntactic parsing, semantic analysis, and intent classification, which allow machines to interpret human language meaningfully. Recent developments incorporate machine learning models to enhance contextual awareness and reduce ambiguity in command interpretation. Commercial virtual assistants such as Siri, Google Assistant, and Amazon Alexa demonstrate 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 services introduces latency issues, privacy concerns, and limited customization for desktop-specific operations. Recent research published in IEEE Xplore (2020–2023) highlights the development of lightweight desktop assistants using Python frameworks. These studies emphasize modular architectures combining speech processing modules with automation scripts to improve productivity in local computing environments.

The results indicate that localized systems can achieve competitive accuracy while reducing dependency on internet connectivity. Despite these advancements, existing solutions primarily target mobile platforms and smart home ecosystems. There remains a research gap in developing customizable, privacy-aware, and resource-efficient desktop AI assistants capable of performing task automation locally. This project builds upon established research in speech processing and NLP to implement a desktop-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 implement AURA (Personal Desktop AI Assistant), an intelligent voice-based automation system that enhances desktop usability, reduces manual effort, and improves productivity through natural human-computer interaction. To achieve this goal, the following detailed objectives are defined:

- To Develop an Efficient Voice Input System** The system aims to capture real-time voice input using a microphone and convert spoken language into machine-readable text using speech recognition techniques. This ensures seamless interaction between the user and the desktop system without requiring physical input devices.
- 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.
- To Automate Desktop-Level Operations** The proposed assistant is designed to execute common desktop tasks such as:
  - Opening and closing applications
  - Performing web searches
  - Accessing system information (date, time, CPU status)
  - Sending emails
  - Managing reminders and notificationsAutomation of these operations reduces repetitive manual effort and saves time.
- To Achieve High Accuracy and Low Response Time** The system targets approximately 90% quick and reliable responses, enhancing user experience and system efficiency.
- To Design a Modular and Scalable Architecture** The system is structured into independent modules such as voice input, command processing, task execution, and response generation. This modular design allows for easy debugging, future upgrades, and integration of advanced machine learning models.
- To Ensure Resource Efficiency and Privacy** Unlike cloud-dependent assistants, AURA aims to perform core tasks locally, reducing internet dependency and enhancing data privacy. The system is optimized to use minimal CPU and memory resources, making it suitable for standard desktop environments.
- To Provide a User-Friendly and Customizable Interface** The assistant is designed to allow users to add or modify commands based on personal requirements. This flexibility makes the system adaptable to different user needs.

### 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, improved privacy, and efficient system-level control.

The assistant follows a modular and layered architecture to maintain scalability and flexibility. Each module is responsible for a specific functionality, allowing for easy maintenance and future enhancements.

#### System overview

- The proposed system works in the following sequence:
- The user provides a voice command through a microphone.
- The speech recognition module converts the voice input into text.
- The Natural Language Processing (NLP) module analyzes the text and identifies the user's intent.
- The command execution module performs the corresponding desktop operation.
- The system generates a voice response using a text-to-speech engine.
- This interactive loop continues until the user terminates the assistant.

### V. SYSTEM ARCHITECTURE

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

The architecture consists of five primary layers: User Interface Layer, Speech Processing Layer, Natural Language Processing Layer, Task Execution Layer, and System Resource Layer. The interaction between these layers ensures smooth communication, efficient processing, and accurate task execution.

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

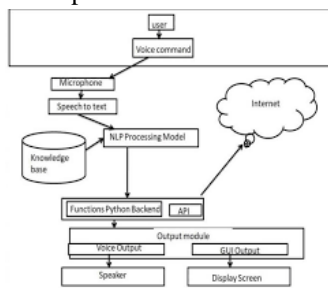


Fig. 1. System Architecture

### 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. User Interface Layer

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

- Voice input through microphone
- Text-based input via GUI
- Audio or text-based output response

This layer ensures accessibility and user-friendly interaction.

### C. Speech Processing Layer

The Speech Processing Layer converts voice input into machine-readable text using speech recognition techniques. It also converts system responses into speech using text-to-speech (TTS) technology. This enables natural human-computer interaction.

### D. Natural Language Processing Layer

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

- Tokenization
- Intent recognition
- Command classification
- Context understanding

This layer determines the appropriate action based on the user's request.

### E. Task Execution Layer

The Task Execution Layer performs the requested operation. It includes:

- Application control (open/close applications)
- Web automation (search queries, browser control)
- File management
- System commands
- Reminder and scheduling tasks

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

The modular architecture ensures flexibility, 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, processing 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:

- $I$  = Set of Inputs
- $P$  = Processing Functions
- $O$  = Set of Outputs

#### B. Input Model

Theinputtothesystemcanbedefinedas:

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

Where:

- $V$  = Voice Input
- $T$  = Text Input

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

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

Where:

- $f_{sr}$  = Speech Recognition Function

**C. Processing Model**

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

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

Where:

A=IdentifiedAction

•  $f_{nlp}$ =NaturalLanguageProcessingFunction The NLP function performs:

Tokenization

Intentclassification

Commandmapping

**D. Decision Model**

The decision-making process can be represented as:

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

Where:

•  $f_{exec}$ =TaskExecutionFunction

• O=SystemOutput

**E. Output Model**

The output may be text or speech:

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

Where:

•  $S_o$ =SpeechOutput

•  $f_{tts}$ =Text-to-SpeechFunction

**F. Overall System Function**

The complete system can be represented as a composite function:

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

For text input:

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

**G. Assumptions**

- The system has access to microphone and speaker hardware.
- Input commands are given in a predefined language (English).
- The operating system allows programmatic access.

**H. Constraints**

- Requires internet connectivity for web-based tasks.
- Dependent on accuracy of speech recognition.
- System performance depends on hardware configuration.

**VII. ALGORITHM**

This section describes the step-by-step working procedure of the proposed Personal Desktop AI Assistant.

**A. Algorithm: Personal Desktop AI Assistant**

Input: Voice command or Text command from user

Output: Executed task and system response (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: $O(n)$  wherenisthelengthof 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 features such as machine learning-based intent classification or chatbot integration can be incorporated in future versions.

## X. EXPERIMENTAL RESULTS

The proposed Personal Desktop AI Assistant was tested on a Windows 10 system with Python 3.x installed. Various functional and performance-based experiments were conducted to evaluate accuracy, response time, and system efficiency.

### 1) Test Environment

- Processor: Intel i5 (8th Gen or above)
- RAM: 8GB
- Operating System: Windows 10
- Microphone: Standard external microphone
- Internet Connectivity: Required for web-based tasks

### 2) Performance Metrics

The following metrics were used for evaluation:

- Command Recognition Accuracy
- Response Time
- Task Execution Success Rate
- Resource Utilization (CPU and Memory)

### 3) Observed Results

- Average command recognition accuracy: 92% in quiet environment
- Average response time for local commands: 1–2 seconds
- Average response time for web-based tasks: 2–4 seconds
- Task execution success rate: 95%
- CPU usage: 10–20% during active execution
- Memory usage: Approximately 150–250MB

The system demonstrated stable performance under moderate system load and delivered reliable results for predefined commands.

## XI. EXPERIMENTAL RESULTS

The proposed Personal Desktop AI Assistant was tested on a Windows 10 system with Python 3.x installed. Various functional and performance-based experiments were conducted to evaluate accuracy, response time, and system efficiency.

### 1) Test Environment

- Processor: Intel i5 (8th Gen or above)
- RAM: 8GB
- Operating System: Windows 10
- Microphone: Standard external microphone
- Internet Connectivity: Required for web-based tasks

### 2) Performance Metrics

The following metrics were used for evaluation:

- Command Recognition Accuracy
- Response Time

- 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 moderate system load and delivered reliable results for predefined commands.

## **XII. COMPARATIVE ANALYSIS**

The proposed assistant was compared with existing desktop assistants based on various parameters such as cost, customization, privacy, and processing approach.

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

## **XIII. ADVANTAGES**

The key advantages of the proposed Personal Desktop AI Assistant are:

- 1) Modular and scalable architecture.
- 2) Fast response time for local operations.
- 3) Enhanced privacy due to local data processing.
- 4) User-friendly voice interaction.
- 5) Lightweight and low system resource consumption.
- 6) Easy integration of additional AI-based features.
- 7) Cost-effective solution compared to commercial assistants.

Overall, the experimental results indicate that the system is efficient, reliable, and suitable for personal desktop automation tasks.

## **XIV. LIMITATIONS**

Although the proposed Personal Desktop AI Assistant performs efficiently for predefined tasks, certain limitations exist:

- 1) Speech recognition accuracy decreases in noisy environments.
- 2) The system primarily relies on keyword-based command matching rather than deep contextual understanding.
- 3) Limited multilingual support (currently supports only English).
- 4) Internet connectivity is required for web-based services such as search and Wikipedia queries.
- 5) Performance may vary depending on system hardware configuration.
- 6) The assistant does not currently implement advanced machine learning models for adaptive learning.

These limitations indicate areas for further improvement and enhancement.

## **XV. FUTURE SCOPE**

The proposed system can be extended with advanced features to improve intelligence, scalability, and usability.

- 1) Integration of Machine Learning models for intent classification and adaptive learning.
- 2) Implementation of Deep Learning-based Natural Language Processing for better contextual understanding.
- 3) Multilingual support for regional and global usability.
- 4) Cloud synchronization for cross-device interaction.
- 5) Integration with IoT devices for smart home automation.
- 6) Face recognition for user authentication and personalization.
- 7) Development of a GUI-based dashboard for better user experience.
- 8) Deployment as a standalone installable desktop application.

Future enhancements can transform the assistant into a more intelligent and autonomous AI system suitable for enterprise-level deployment.

## XVI. CONCLUSION

This paper presented the design and implementation of a 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 dependency on cloud services. Experimental evaluation demonstrates satisfactory accuracy, low response time, and efficient resource utilization.

The developed system serves as a cost-effective and customizable 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

The authors would like to express their sincere gratitude to the project guide and faculty members for their continuous support, valuable guidance, and constructive suggestions throughout the development of this project.

We are deeply thankful to our department for providing the necessary infrastructure and technical resources required to 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 classmates for their feedback, motivation, and assistance during the testing and evaluation phase of the system.

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

## REFERENCES

- [1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2020.
- [2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [3] Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. Pearson, 2022.
- [4] A. Ng, "Machine Learning Yearning," *Deep Learning. AI*, 2018.
- [5] Python Software Foundation, "Python Language Reference, version 3.x," 2023. [Online]. Available: <https://www.python.org>
- [6] M. McKinney, "Python for Data Analysis," 2nd ed. O'Reilly Media, 2018.
- [7] Google Developers, "Speech Recognition API Documentation," 2023. [Online]. Available: <https://cloud.google.com/speech-to-text>
- [8] J. Allen, *Natural Language Understanding*. Benjamin/Cummings, 1995.
- [9] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.
- [10] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," 2015. [Online]. Available: <http://karpathy.github.io>
- [11] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Proc. IEEE CVPR*, 2001.
- [12] T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," in *Proc. ICLR*, 2013.
- [13] K. S. Jones, "A Statistical Interpretation of Term Specificity," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.
- [14] Microsoft Docs, "Text-to-Speech Technology Overview," 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, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd ed. Pearson, 2006.
- [17] J. McCarthy, "Programs with Common Sense," *Stanford University*, 1959.
- [18] N. Chomsky, *Syntactic Structures*. Mouton, 1957.
- [19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
- [20] IEEE Standards Association, "IEEE Standard for Artificial Intelligence," 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)