VOICE ASSISTANCE USING ARTIFICIAL INTELLIGENCE

A Project Report submitted in thepartialfulfillment of the Requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATIONTECHNOLOGY

Submitted by

M.V.SAI KALYAN (21471A1236)

Mr. T. VENKATA MANOHAR, M.Tech.

Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

NARASARAOPETAENGINEERING COLLEGE: NARASAROPET (AUTONOMOUS)

Accredited by NAAC with A+ Grade, ISO-9001:2015 certified, Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada

KOTAPPAKONDA ROAD, YELAMANDA VILLAGE, NARASARAOPET -522601

2024-2025

NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET

(AUTONOMOUS)

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project work entitled "Voice Assistance Using Artificial Intelligence" is a bonafide work done by M.V.Sai Kalyan (21471A1236) partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in the Department of **INFORMATION TECHNOLOGY** during academic year **2024-25**.

PROJECT GUIDE	**PROJECT CO ORDINATOR**
T. V.Manohar, M.Tech	**Dr. Shaik Mohammad Jany,**M.Tech.,Ph.D.
Asst. Professor	Asst. Professor

HEAD OF THE DEPARTMENT	**EXTERNALEXAMINR**
Dr. B. Jhansi Vazram, B.Tech., M.Tech., Ph.D.	
Professor& HOD	

Viva Voce Date: ………………….

DECLARATION

We, hereby declare that the project entitled **"Voice Assistance Using Artificial Intelligence"** is an outcome of our efforts which has been duly acknowledged. We also declare that it has not been previously or currently submitted for any other institutions or universities. This project is done under the guidance of Mr . **T. Venkata Manohar, M.Tech Asst. Professor**, Department of Information Technology, **NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS),** is is submitted in the partial fulfillment of the requirements for the award of a degree in **INFORMATION TECHNOLOGY**.

By

M. V. Sai Kalyan 21471A1236

ACKNOWLEDGEMENTS

We wish to express our thanks to the various personalities who are responsible for the completion of the project. We are extremely thankful to our beloved chairperson **Sri M.V. Koteswara Rao, B.Sc.,** who took a keen interest in us in every effort throughout this course. We our sincere gratitude to our beloved principal **Dr. S. Venkateswarlu, Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

We express our heartful gratitude to **Dr. B. Jhansi Vazram, B.Tech., M.Tech., Ph.D.,** Head of the Department of Information Technology and our project guide **Mr. T. Venkata Manohar, M.Tech**, Assistant professor for his invaluable guidance, continuous support, and constant encouragement, which enabled us to successfully complete our project on time.

We extend our sincere thanks to **Dr. Sk. Mohammed Jany, M.tech, Ph.D. ,** Assistant professor & Project coordinator of the project for extending her encouragement. Their profound knowledge and willingness have been a constant source of inspiration for us throughout this project work.

We extend our sincere thanks to all other teaching and non-teaching staff in the department for their cooperation and encouragement during our B. Tech degree. We have no words to acknowledge the warm affection, constant inspiration, and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who were involved in giving valuable suggestions had clarifying out doubts which had really helped us in completing our project.

By

M. V. Sai Kalyan 21471A1236

NARASARAOPETA ENGINEERING COLLEGE (AUTONOMOUS)

INSTITUTE VISION AND MISSION

INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student- centric teaching-learning practices as well as research for the transformation of lives and communities.

INSTITUTION MISSION

M1: Provide the best class infra-structure to explore the field of engineering and research.

M2: Build a passionate and a determined team of faculty with student centric teaching, imbibing experiential, innovative skills.

M3: Imbibe lifelong learning skills, entrepreneurial skills and ethical values in students for addressing societal problems.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

INSTITUTE VISION AND MISSION

DEPARTMENT VISION

To transform into a research and technological hub to develop prominent IT professionals to serve the needs of industry and society.

DEPARTMENT MISSION

The department of Information Technology is committed to

M1: Induce preliminary and contemporary IT principles of the industry among the students.

M2: Develop strong force of students to solve the real time problems of the IT industry.

M3: Incubate the students with emerging entrepreneur intelligence.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Ability to analyze and develop computer programs in the areas related to Algorithms, system software, application software, web design, big data analytics, database design and networking for efficient design of computer-based systems of varying complexity.

PSO2: Design, Implement and evaluate a computer-based system to meet desired needs.

PSO3: Develop IT applications services with help of different current engineering tools.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The graduates of the programmer are able to:

PEO1: Apply the knowledge of Mathematics, Science and Engineering fundamentals to identify and solve Information Technology problems.

PEO2: Use various software tools and technologies to solve problems related to academia, industry and society.

PE03: Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team member with continuous learning.

PEO4: Pursue higher studies and develop their career in software industry.

NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

PROGRAM OUTCOMES (POS)

PO1: Engineering Knowledge: Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization to develop to the solution of complex engineering problems.

PO2: Problem Analysis: Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development.

PO3: Design/Development of Solutions: Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required.

PO4: Conduct Investigations of Complex Problems: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions.

PO5: Engineering Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems.

PO6: The Engineer and The World: Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment.

PO7: Ethics: Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. **PO8: Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication: Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.

PO10: Project Management and Finance: Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning: Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

NARASARAOPETA
NEC ENGINEERING COLLEGE
(AUTONOMOUS)

PROJECT WORK COURSE OUTCOMES (COS)

CO421.1: Analyze the System of Examinations and identifythe problem.

CO421.2: Identify and classify the requirements. **CO421.3:** Review the Related Literature.

CO421.4: Design and Modularize the project

CO421.5: Construct, Integrate, Test and Implement the Project.

CO421.6: Prepare the project Documentation and present the Report using appropriate method.

CourseOutcomes – Program Outcome correlation:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO1	PO1	PSO	PSO	PSO3
C421.1	2	3										2		
C421.2			2		3							2		
C421.3				2		2	3					2		
C421.4			2			1	2					3	2	
C421.5					3	3	2	3	2	2	1	3	2	1
C421.6								3	2	1		2	3	

Note: The values in the above table represent the level of correlation between COs and POs:

1. Low level
2. Medium level
3. High level

Project work mapping with various courses of Curriculum with Attained POs:

Name of the course from which principles are applied in this project	**Description of thedevice**	**Attained PO**
R20C2204, R20C22L3	Gathering the requirements and defining the problem, plan to develop a model for Hate speech detection using BERT	PO1, PO3
R20CC421,R20C2204, R20C22L3	Each and every requirement i critically analyzed, the process mode is identified	PO2, PO3
R20CC421,R20C2204, R20C22L3	Logical design is done by using the unified modelling language which involves individual team work	PO3, PO5, PO9
R20CC421,R20C2204, R20C22L3	Each and every module is tested, integrated, and evaluated in our project	PO1, PO5
R20CC421,R20C2204, R20C22L3	Documentation is done by all our Three members in the form of a group	PO9
R20CC421,R20C2204, R20C22L3	Each and every phase of the work in a group is presented periodically	PO9, PO10
R20C2202,R20C2203, R20C1206,R20C3204, R20C4110	Implementation is done and the project will be handled by the social media users and in the future updates in our project can be done based on the detection of tweets	PO4
R20C32SC4.3	The physical design includes website check whether an tweet is hate or non hate	PO5, PO6

ABSTARCT

The main goal of Artificial intelligence (AI) is the realization of natural dialogue between humans and machines. We use python as a programming language because it have a major libraries which is use to execute commands. By using python installer packages our personal virtual assistant recognizes the user voice and process on it. Voice assistants are the great innovation in the field of AI that can change the way of living of the people in a different manner.. Initially, the voice assistant was mostly being used in smartphones and laptops but now it is also coming as home automation and smart speakers. Many devices are becoming smarter in their own way to interact with human in an easy language. The Desktop based voice assistant are the programs that can recognize human voices and can respond via integrated voice system. This paper will define the working of a voice assistants, their main problems and limitations. In this paper it is described that the method of creating a voice assistant without using cloud services, which will allow the expansion of such devices in the future.

Keywords: Voice assistants, text-to-speech, speech recognition, chatbots, and static voice assistants

INDEX

LIST OF FIGURES

1. INTRODUCTION

1.1 INTRODUCTION

Virtual assistant is used to run machine like laptop or PC's on your own command. Virtual assistant is an application program that understands natural language and voice commands to complete tasks for the users. The Users can ask their assistants' questions, control home automation devices, and media playback via voice, and manage other basic tasks such as email, to-do lists, open or close any application, send messages to anyone on whatsapp etc. with verbal commands only. Some other types of Voice Assistant are:

- Intelligent Personal Assistant
- Automated Personal Assistant
- Virtual Digital Assistants
- Chat bot

Nowadays virtual assistant is very useful to human. It makes human life easier like operate PC's or laptop on only voice command. Virtual assistant is a less time consuming. By using virtual assistant we saves our time and contribute in other works. Virtual assistants are typically cloud-based program that requires internet connected devices. Virtual assistant is the flexibility to contract for just the services they need. For creating virtual assistant for your computer go from basics python. Virtual assistants are task-oriented.

Virtual assistant's ability to understand and perform requests. Virtual assistants is a software that understands verbal and written commands and completes task assigned by clients. Virtual assistants are able to interpret human speech and respond via synthesized voices.

There are several voice assistants in market like Siri for apple TV remote, Google Assistant for pixel XL smartphones, Alexa as a smart speaker which is developed by using Raspberry Pi, Microsoft Cortana for windows 10.

As like this all virtual assistants we also created a virtual assistant for windows. We use Artificial Intelligence technology for this project. Also use python as a programming language, because python offers a good major libraries. For this software use microphone as input device to receive voice requests from user and speaker as output device to give the output voice. This process is the combination of several different technologies like voice recognition, voice analysis and language processing. Virtual assistant use Natural Processing language to match user text or voice input to executable commands. When a user give a command to personal virtual assistant to perform a task, the natural language is converted the audio signals into digital signals Virtual assistants can provide several services which includes,

- Showing weather condition.
- Scheduling appointment.
- Showing date-time.
- Managing emails.
- Open apps.

Over time, advancements in natural language processing (NLP), machine learning, and computational linguistics motivated the seamless human-computer interaction, researchers and developers have embraced the challenge of building AI voice assistants that not only understand spoken language but also respond intelligently and adaptively to user queries and commands. At the heart of every AI voice assistant lies a complex ecosystem of technologies working

Voice assistants are one of the most practical and popular applications of artificial intelligence in every day life. These AI-Powered systems are designed to understand spoken language, interpret user intent ,and respond with helpful actions or information all through natural, conversational interaction.

Commonly found in smartphones ,smart speakers, cars and customer service systems, voice assistants like Siri ,Alexa, Google Assistant and Cortana have transformed how people interact with technology. They allow users to perform tasks hands-free, such as setting alarms, sending messages ,playing music or controlling smart home devices all with simple voice commands.

Challenges in building voice Assistance using Artificial Intelligence:

- **Latency:** Minimizing the delay between speech and response.
- **Cross-platform support:** Making the assistant work on different devices and apps.

In Automatic Speech Recognition we have the challenges like accents and dialects , Background Noise,Low-resource language. In Natural language we have the challenges like Ambiguity in language,context management Domain adaptation and finally Intent classification errors. In Dialogue management we have the challenges like stack tracking personalization and Error handling. In text –to -Speech we have the challenges like Naturalness and expressiveness ,Emotion and tone control , Real-time processing.

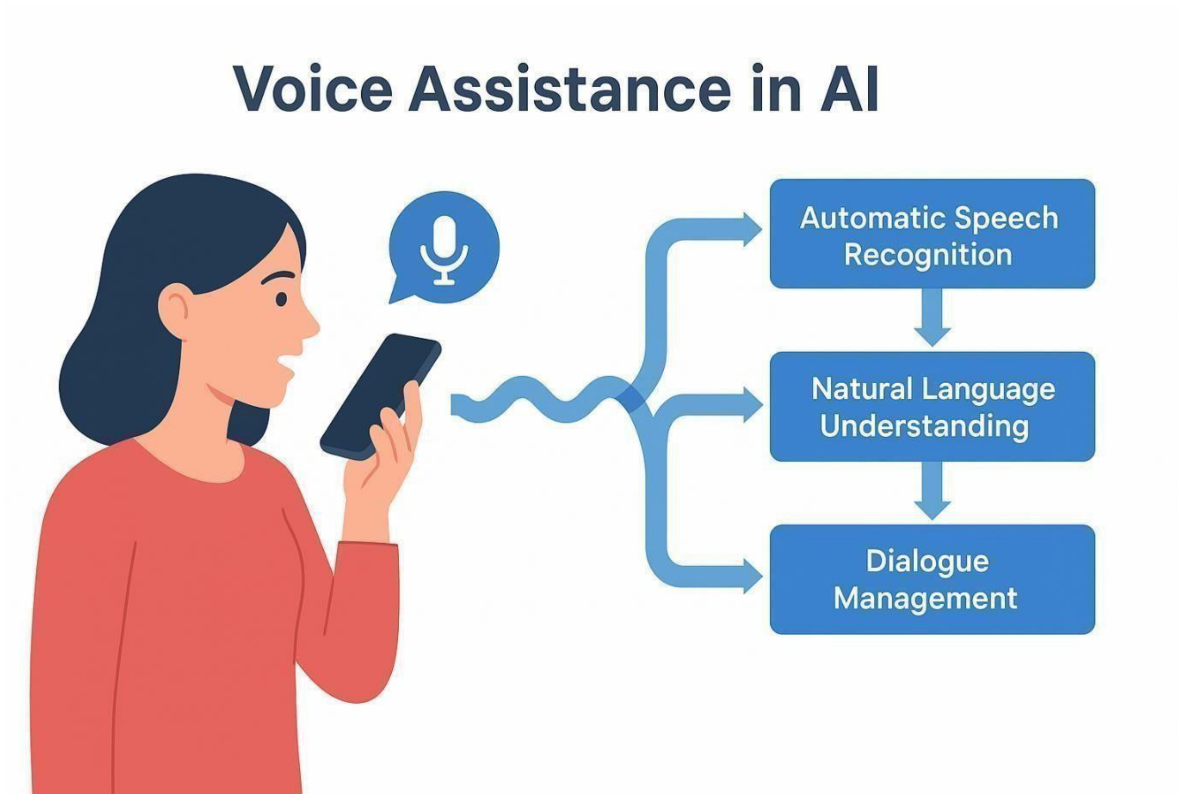


Fig 1.1 Voice Assistance in AI

1.2 EXISTING SYSTEM

- The system will keep listening for commands and the time for listening is variable which can be changed according to user requirements.
- If the system is not able to gather information from the user input it will keep asking again to repeat till the desired no. of times.
- The system can have both male and female voices according to user requirements.
- Features supported in the current version include playing music, emails, texts, search on Wikipedia, or opening system installed applications, opening anything on the web browser, etc.

[User Speech]

↓

[ASR: Deep Speech / Alexa ASR Engine]

↓

[NLU: Intent Detection (BERT-style model)]

↓

[Dialogue Manager: FSM + RL]

↓

[Response Generation]

↓

[TTS: Neural (Amazon Polly)]

↓

[Spoken Response]

These are the existing ways that we would use in the every of the voice assistance using artificial intelligence or in other way. From the Existing System we have the different types of disadvantages that we have them following as well.

1.2.1 DISADVANTAGES OF EXISTING SYSTEMS

Despite significant advancements in AI-Powered voice assistance ,current systems still face Several limitations and challenges. These disadvantages affect usability performance and trustworthiness key drawbacks include:

- .Limited Context Understanding: Most assistants struggle with maintaining long- term conversation context .They often fail in multi-turn dialogue or complex follow -up questions

- Language and Accent Barriers: Accuracy .drops with non-standard accents, regional dialects, or code switching. Low resources languages are often unsupported or poorly served.

- Dependence on internet connectivity: Many questions require a constant online connection to access cloud-based models. Offline functionality is limited or non existent in most systems.

- Privacy and Data security concerns: Continuous listening device raise user concerns about surveillance and data misuse. Some systems store voice data, which can be vulnerable to breaches or misuse.

The system will keep listening for commands and the time for listening is variable which can be changed according to user requirements.

1.3 PROPOSED SYSTEM

The proposed system is an advanced version of existing open-source voice assistants, aiming for superior functionality, precision, accuracy, and security. It targets accuracy levels akin to Google Assistant and security standards similar to Apple's. It swiftly analyzes input data, transcribes speech into text, and displays it on the screen, utilizing the Google Speech Recognition package in Python. The interface facilitates speech input, capturing audio via the built-in microphone and relaying recognized words for processing. Upon verification, accurate words trigger dedicated Python child processes to execute system commands. The resulting data is displayed for user confirmation before refreshing for the next input, ensuring seamless interaction. Designed with user-friendly instructions and animations, the application prioritizes user engagement. Importantly, it doesn't store user data, mitigating common security risks, and is compatible with personal computers across various operating systems.

With the rapid development of artificial intelligence and deep learning, next-generation voice assistant systems aim to overcome the limitations of existing models. These proposed systems offer several improvements that enhance usability, personalization, and performance.

Enhanced Context Awareness:

- Advanced AI models (like GPT-4 or dialogue transformers) allow the assistant to understand and maintain multi-turn conversation context. This enables more natural and human-like interactions. Advantage- Accurate results with proper security for the user.

2. LITERATURE SURVEY

Cohen (1960) [1] introduced a statistical measure known as the **coefficient of agreement for nominal scales**. This metric was designed to assess the level of agreement between two raters who classify items into mutually exclusive categories. It accounts for the possibility of the agreement occurring by chance, making it more robust than simple percentage agreement.

This coefficient, often referred to as **Cohen's Kappa**, has since become a standard tool in research fields where categorical data is evaluated. Its application spans psychology, education, medical research, and many other domains where inter-rater reliability is essential. The measure provides a quantifiable way to determine consistency between evaluators, which is particularly useful in subjective assessments.

Given its strength in measuring agreement, Cohen's Kappa could potentially be applied to evaluate the performance and reliability of **voice assistant systems**. For example, it could be used to compare how consistently a voice assistant system interprets user commands against a human- annotated standard. This would provide insight into the system's accuracy and consistency in understanding and responding to user inputs.

Hinton et al. (2012) [2] made a groundbreaking contribution to the field of speech recognition by introducing the use of **deep neural networks (DNNs) for acoustic modeling**. This work represented a significant departure from traditional approaches that relied heavily on Gaussian Mixture Models (GMMs), showcasing how DNNs could model complex patterns in speech data more effectively.

The study brought together insights from four major research groups, highlighting the shared understanding of how deep learning techniques could improve the accuracy and robustness of speech recognition systems. Their findings demonstrated that DNNs could outperform previous models by learning hierarchical feature representations directly from raw input data, enabling better generalization across various acoustic conditions.

This foundational research has played a critical role in the development of modern **voice assistant technologies**, such as Siri, Alexa, and Google Assistant. The principles and techniques introduced in this work continue to inform the design of speech processing systems, making voice-based interaction more natural, responsive, and accurate for users.

Young et al. (2019) [3] offered a **comprehensive overview of recent trends in deep learning- based natural language processing (NLP)**. Their work highlighted how deep learning models, particularly neural networks, have significantly advanced the field by enabling machines to better understand, generate, and interact with human language. This shift has transformed NLP from rule- based and statistical methods to more dynamic, data-driven approaches.

The study covered key developments such as the use of word embeddings, recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformers. These innovations have not only improved tasks like sentiment analysis, machine translation, and question answering but also paved the way for **more context-aware and semantically rich language models**, which are essential for natural and engaging human-computer interaction.

These advancements are particularly relevant to the development of **voice assistant systems**, where effective dialogue generation and understanding are crucial. By leveraging deep learning-based NLP models, voice assistants can better interpret user intents, maintain context across interactions, and generate more coherent and relevant responses, enhancing the overall user experience.

Chorowskietal. (2015) [4] introduced **attention-based models for speech recognition**, marking a significant advancement in how systems process and interpret spoken language. Traditional models often struggled with aligning audio input with corresponding text output, particularly in longer sequences. Attention mechanisms addressed this challenge by allowing the model to focus dynamically on different parts of the input when generating each element of the output.

Their approach built upon earlier sequence-to-sequence models by integrating attention mechanisms, which helped improve both **accuracy and efficiency** in speech recognition tasks. These models could better manage variable-length inputs and maintain contextual awareness throughout the decoding process, significantly improving recognition performance in complex scenarios.

The impact of this work is particularly evident in modern **voice assistant technologies**, where fast and accurate speech processing is essential. By enabling more precise alignment between speech and text, attention-based models enhance the responsiveness and reliability .

Li et al. (2016) [5] explored the application of **deep reinforcement learning (DRL) for dialogue generation**, presenting a novel approach to developing more interactive and goal- oriented conversational agents. Unlike traditional supervised learning models that rely on static datasets, DRL enables systems to learn optimal dialogue strategies through trial and error, guided by reward signals.

The study introduced a framework in which the dialogue model is treated as an agent interacting with a user in a simulated environment. By optimizing long-term rewards—such as maintaining coherent conversations, achieving specific dialogue goals, or enhancing user satisfaction—the model can learn to generate more contextually appropriate and engaging responses over time.

This research has significant implications for **voice assistant systems**, which rely heavily on fluid and meaningful dialogue with users. By incorporating deep reinforcement learning, voice assistants can move beyond simple scripted responses and develop more dynamic conversational abilities, ultimately leading to a more personalized and intelligent user experience.

Kim et al. (2021) [6] presented **a comprehensive review of voice-based virtual personal assistants (VPAs)**, highlighting the evolution, capabilities, and limitations of these systems. Their work examines the key technologies that power VPAs, including automatic speech recognition (ASR), natural language understanding (NLU), and text-to-speech (TTS) synthesis, all of which are essential for enabling effective human-computer interaction.

The review delves into various architectures and design approaches used in popular VPAs such as Siri, Alexa, and Google Assistant. It also explores how advancements in deep learning and AI have improved these systems' ability to understand user context, personalize responses, and handle multi-turn conversations. Despite these advancements, the authors point out ongoing challenges such as privacy concerns, handling ambiguous user input, and ensuring inclusivity across diverse accents and languages.

By providing a broad and detailed overview, Kim et al. (2021) contribute valuable insights into the current **state-of-the-art techniques and persistent challenges** in voice assistant development. Their findings serve as a useful resource for researchers and developers aiming to enhance the functionality, reliability, and user experience of future voice-based virtual assistants.

Kim et al. (2021) [8] conducted **a comprehensive review of voice-based virtual personal assistants (VPAs)**, focusing on the technological foundation, evolution, and emerging trends in the field. Their study outlines the core components of VPAs, such as automatic speech recognition (ASR), natural language understanding (NLU), dialogue management, and text-to- speech (TTS) synthesis. These components work together to facilitate seamless human-machine communication, making VPAs an integral part of modern smart devices.

The review highlights the **rapid advancements in artificial intelligence and deep learning** that have significantly enhanced the capabilities of VPAs. Kim et al. discuss how modern systems are increasingly capable of understanding user context, maintaining conversational continuity, and adapting responses to suit individual preferences. They also compare the performance and design strategies of well-known VPAs like Apple's Siri, Amazon's Alexa, and Google Assistant, offering a broad perspective on current industry standards.

Despite notable progress, the study identifies **several key challenges** that still hinder the full potential of voice-based assistants. These include limitations in understanding natural, unstructured speech, ensuring privacy and security, and addressing the diverse linguistic needs of global users. By outlining both the achievements and the ongoing challenges, Kim et al. provide a valuable roadmap for future research and development in the field of voice assistant technologies.

Møller (1993) [9] introduced **a scaled conjugate gradient (SCG) algorithm** designed to accelerate the process of supervised learning in neural networks. This method was developed as an alternative to traditional backpropagation and other second-order optimization techniques, offering a more efficient approach for training feedforward neural networks.

The SCG algorithm combines the model accuracy of second-order methods with the computational efficiency of first-order methods. Unlike standard conjugate gradient methods, it avoids the need for time-consuming line searches, making it particularly well-suited for large- scale learning problems. Møller's approach balances speed and performance by scaling the gradient updates based on curvature information.

"Speech Command Recognition Using Convolutional Neural Networks." Applied Sciences, 9(20), 432

Weng et al. (2019) [10] investigated the use of **convolutional neural networks (CNNs)** for speech command recognition, a key component in the functionality of voice assistant systems. Their work focused on designing models capable of accurately identifying short speech commands, which are commonly used in real-world human-computer interaction.

The study demonstrated how CNNs, typically used in image recognition, could be effectively applied to spectrogram representations of audio data. By treating these spectrograms as two- dimensional input, the CNNs learned spatial features that are crucial for distinguishing between different spoken commands, even in noisy environments.

This research has strong implications for **improving the responsiveness and accuracy** of voice- activated systems. Efficient speech command recognition allows voice assistants to reliably understand user instructions, contributing to a more intuitive and seamless user experience in applications ranging from smart homes to mobile devices.

Graves, Mohamed, and Hinton (2013) [11] introduced the use of **deep recurrent neural networks (RNNs) for speech recognition**, marking a significant shift in the design of acoustic models. Their work explored how RNNs, particularly those with Long Short-Term Memory (LSTM) units, could effectively model the temporal dynamics of speech data, outperforming traditional feedforward networks.

One of the key advantages of RNNs highlighted in the study is their ability to **retain contextual information over time**, which is crucial for recognizing patterns in sequential data like speech. The authors demonstrated that deep RNNs could learn long-term dependencies and adapt to varying input lengths, making them ideal for handling continuous speech in real-time applications.

This research has had a profound impact on the development of modern **voice assistant technologies**, where understanding spoken language accurately and in context is essential. By leveraging deep RNNs, systems like Google Assistant and Siri have been able to achieve greater accuracy in speech recognition, contributing to more natural and fluid human-computer interactions.

3. SYSTEM REQUIREMENTS

The software requirement specification can produce at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by established a complete information description, a detailed functional description, a representation of system behavior, and indication of performance and design constrain, appropriate validate criteria, and other information pertinent to requirements.

Regarding software compatibility, the application is designed to run smoothly on Windows10. Python serves as the primary coding language due to its flexibility and simplicity, facilitating development tasks. To expedite the development process, users are recommended to employ Python distributions like Anaconda and Flask. Additionally, accessing the application's user interface is feasible through any up-to-date browser like Chrome. These software prerequisites guarantee adaptability and performance across different systems, ensuring users can efficiently leverage the application.

3.1 Software Requirements:

- Operating system: Windows
- Code Technology: Python, Node.js
- API: Google Speech Recognition API

3.2 Hardware Requirement:

- Processor: Intel i3
- RAM: 4 GB
- Hard Disk: 256 GB

4. SYSTEM ANALYSIS AND DESIGN

4.1 METHODOLOGY:

4.1.1 User

- The user interacts with the system solely through voice commands. There is no provision for prompt input.

4.1.2. Monitor:

- The Monitor component is responsible for managing the recognition and processing of voice commands from the user.
- It continuously listens for spoken commands through the microphone, utilizing speech recognition technology to convert the spoken words into text format.
- Upon recognizing a command, the Monitor component immediately proceeds to execute the corresponding system task without waiting for user validation, assuming a high level of confidence in the accuracy of the transcription.
- After completing the task, the Monitor component displays the output or response generated by the system, providing feedback to the user.

4.1.3. Microphone:

- The Microphone component remains unchanged and continues to serve as the input device for capturing spoken commands from the user.
- It listens to the user's voice and transmits the audio signals to the Monitor component for processing algorithms:

4.2 BLOCK DIAGRAM

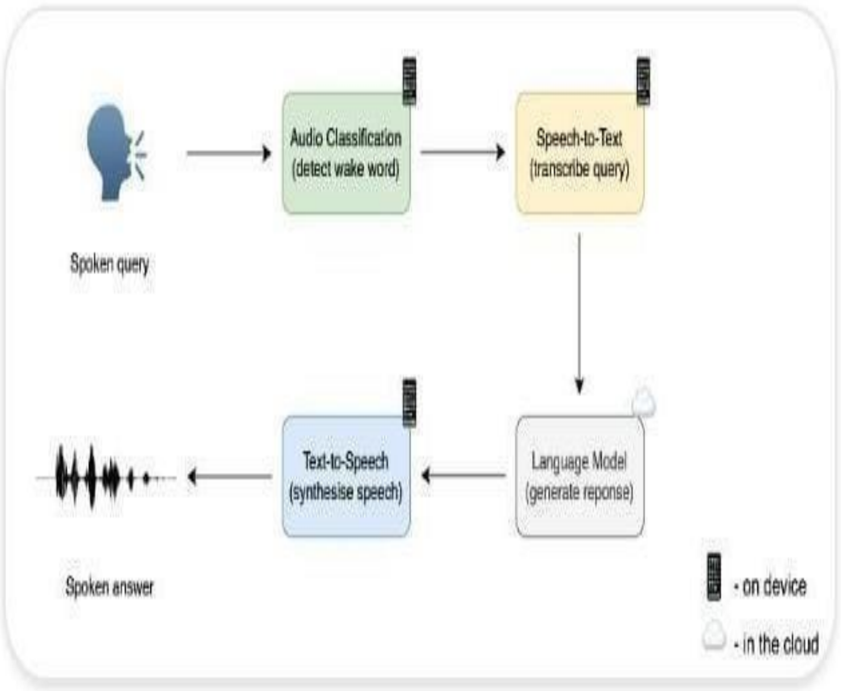

Spoken query
Audio Classification (detect wake word)
Speech-to-Text (transcribe query)
Language Model (generate reponse)
Text-to-Speech (synthesise speech)
Spoken answer
- on device
- in the cloud

Fig 4.2 Block diagram of AI Voice Assistant

Above block diagram illustrates-

User:

1) Give prompt input
2) Give voice commands

Monitor:

1) Recognize speech and convert to text
2) Accept user validation
3) Performs system task
4) Display output

Microphone:

1) Listen to spoken commands

4.3 USE CASE DIAGRAM

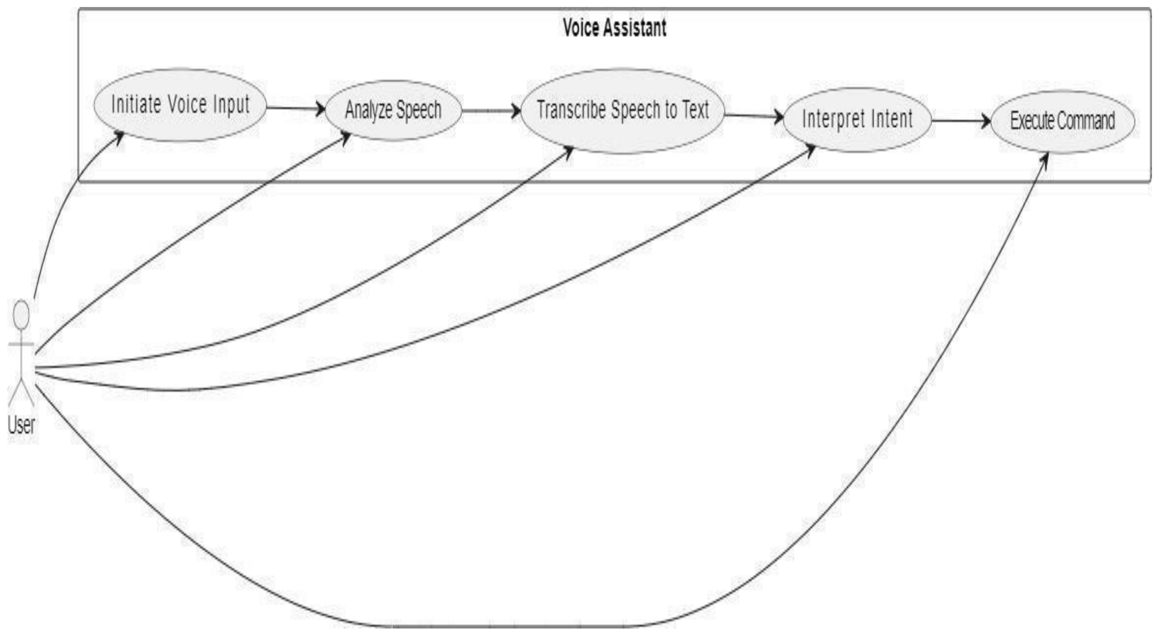


Fig 4.3 Use case diagram of AI Voice Assistant

<u>User (U)</u>: The primary actor initiating interactions with the Voice Assistant system. **<u>Voice Assistant (rectangle):</u>** Represents the Voice Assistant system, which consists of several use cases:

- Initiate Voice Input (IVI): This use case involves the user triggering the Voice Assistant by providing voice input.

- Analyze Speech (AS): Once voice input is initiated, the system analyzes the speech to identify and extract relevant information.

- Transcribe Speech to Text (TST): After analyzing the speech, the system transcribes it into text format for further processing.

- Interpret Intent (II): The transcribed text is then interpreted to determine the user's intent or command.

- Execute Command (EC): Based on the interpreted intent, the system executes the corresponding command or action.

Interactions (Arrows): Arrows indicate the flow of interactions between the user and the Voice Assistant system as well as the sequential flow between different use cases within the system. For example, the user triggers voice input, which leads to speech analysis, transcription, intent interpretation, and command execution in a sequential manner.

4.4 CLASS DIAGRAM

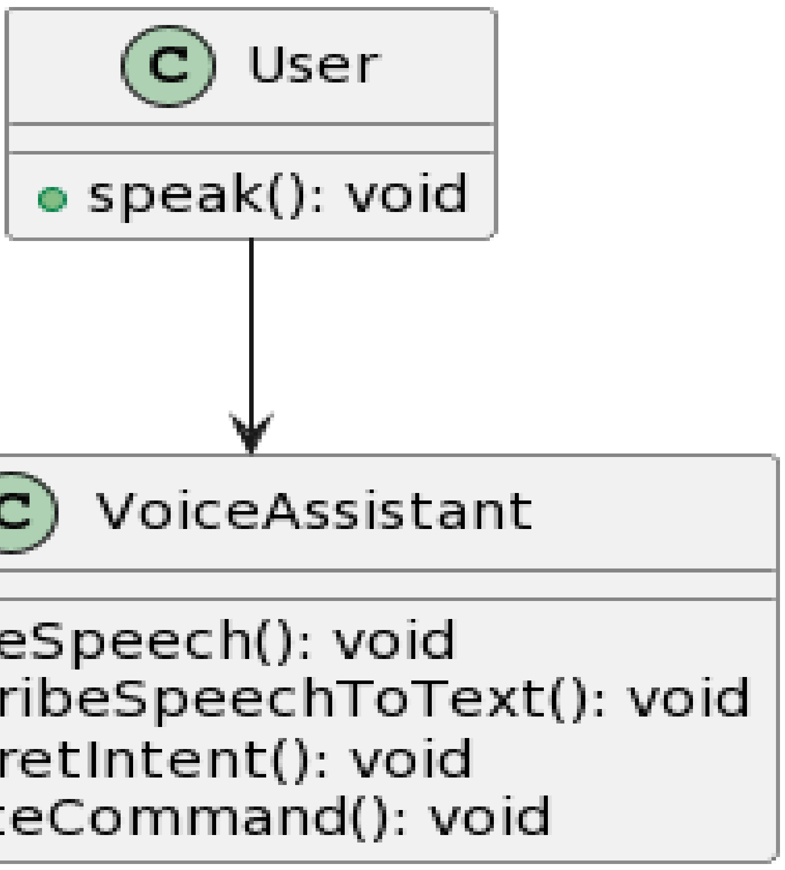


Fig 4.4 Class diagram of AI Voice Assistant

<u>User Class:</u>

Attributes: - None Methods:

speak(): void: Represents the method for the user to speak and interact with the Voice Assistant system. This method triggers the voice input process.

<u>VoiceAssistant Class:</u>

Attributes:- None Methods:

analyzeSpeech(): void: This method is responsible for analyzing the speech input received from the user. It performs tasks such as identifying keywords, detecting language, and extracting relevant information.

transcribeSpeechToText(): void: Once the speech is analyzed, this method transcribes it into text format. It converts spoken words into a readable and processable text representation. interpretIntent(): void: After transcribing the speech, this method interprets the user's intent or command based on the transcribed text. It identifies the user's purpose or desired action from the input.

executeCommand(): void: Finally, this method executes the command or action corresponding to the interpreted intent. It performs tasks such as retrieving information, controlling devices, or providing responses to the user.

<u>Interactions:</u>

The arrow indicates the relationship between the "User" class and the "VoiceAssistant" class, representing the interaction flow where the user triggers the Voice Assistant system by speaking, leading to the execution of various methods within the "VoiceAssistant" class.

4.5 SEQUENCE DIAGRAM

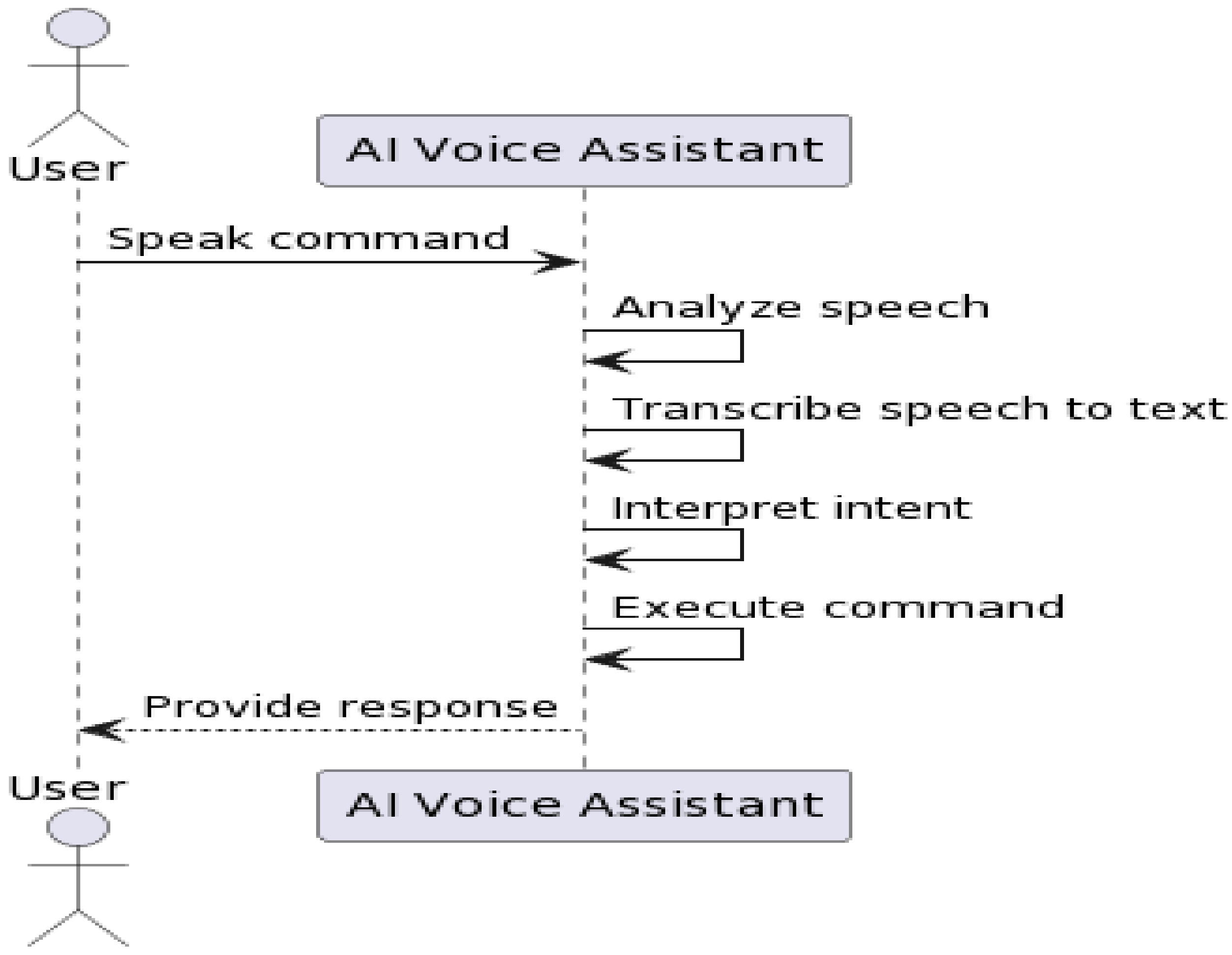


Fig 4.5 Sequence diagram of AI Voice Assistant

User: Initiates the interaction by speaking a command or query to the Voice Assistant system.

AI Voice Assistant: Receives the spoken command from the user and proceeds with the following steps:

Analyze speech: The system analyzes the incoming speech input to identify and understand the spoken words, detecting any language variations or accents.

Transcribe speech to text: After analyzing the speech, the system transcribes it into text format, converting the spoken words into a readable form for processing.

Interpret intent: Based on the transcribed text, the system interprets the user's intent or command, determining the action to be taken or the information to be retrieved.

Execute command: With the intent understood, the system executes the corresponding command or action, performing tasks such as retrieving information from databases, controlling connected devices, or providing responses to the user's query.
AI Voice Assistant: After executing the command, the system generates a response or performs the requested action.
User: Receives the response from the Voice Assistant, completing the interaction loop.

4.6 ACTIVITY DIAGRAM

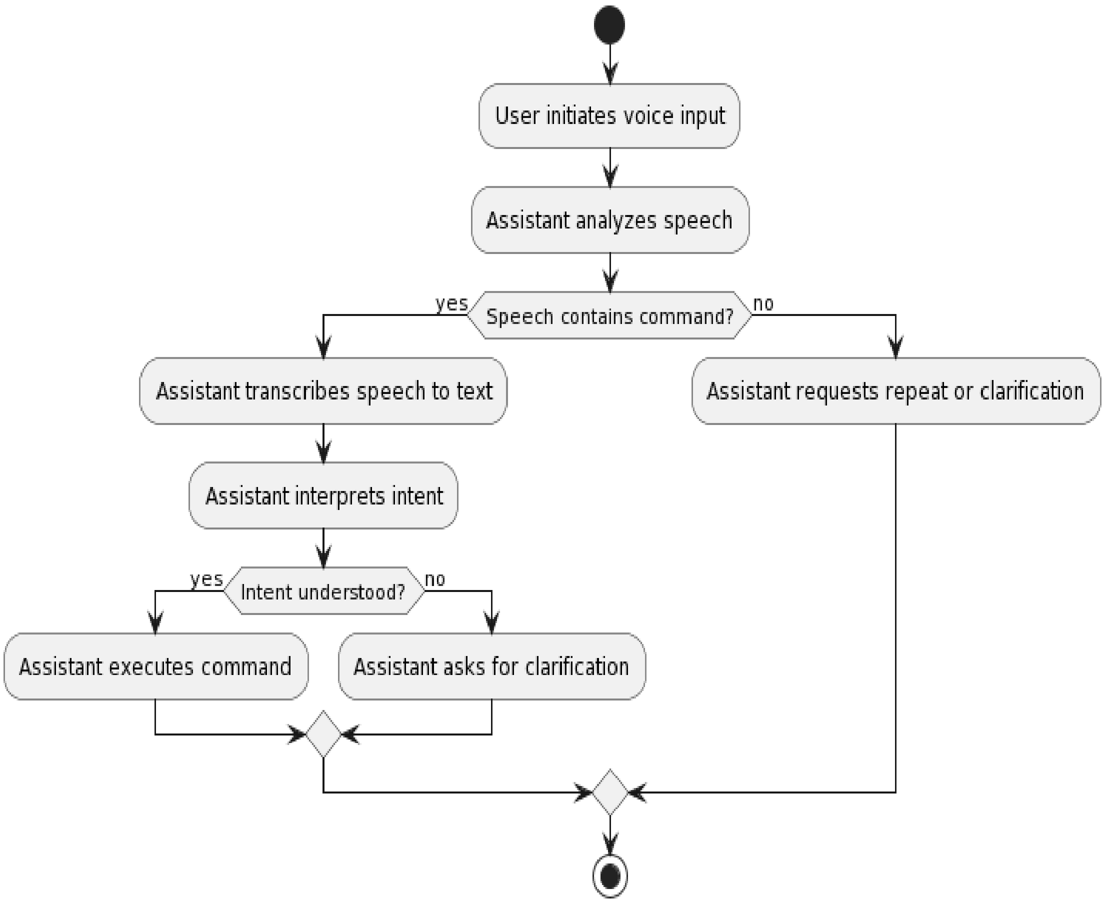

User initiates voice input
Assistant analyzes speech
yes
Speech contains command?
no
Assistant transcribes speech to text
Assistant requests repeat or clarification
Assistant interprets intent
yes
Intent understood?
no
Assistant executes command
Assistant asks for clarification

Fig 4.6 Activity diagram of AI Voice Assistant

User Interaction: The user initiates the interaction by speaking a command or query to the Voice Assistant.

Voice Assistant Processing: The Voice Assistant systemreceives the user's input and proceeds withthe following activities:

Initiate Voice Input: The system begins processing the user's spoken input.

Analyze Speech: The system analyzes the incoming speech data, identifying patterns, and characteristics to understand the spoken words effectively.

Transcribe Speech to Text: After analyzing the speech, the system converts the spoken words into text format, facilitating further processing.

Interpret Intent: Based on the transcribed text, the system interprets the user's intent or command, determining the action to be taken.

Execute Command: With the intent understood, the system executes the corresponding command or action, such as retrieving information, performing tasks, or providing responses.

Response Delivery:

Upon executing the command, the Voice Assistant generates a response or performs the requested action.

Completion:

The interaction loop is completed as the Voice Assistant provides the response back to the user, concluding the activity.

4.7 PACKAGES:

- Speech recognition — Speech recognition is an important feature used in house automation and in artificial intelligence devices. The main function of this library is it tries to understand whatever the humans speak and converts the speech to text.
- pyttsx3 — pyttxs3 is a text to speech conversion library in python. This package supports text to speech engines on Mac os x, Windows and on Linux.
- wikipedia — Wikipedia is a multilingual online encyclopedia used by many people from academic community ranging from freshmen to students to professors who wants to gain information over a particular topic. This package in python extracts data’s required from Wikipedia..
- ecapture — This module is used to capture images from your camera
- datetime — This is an inbuilt module in python and it works on date and time.
- os — This module is a standard library in python and it provides the function to interact with operating system.
- time — The time module helps us to displaytime.
- Web browser — This is an in-built package in python. It extracts data from the web.
- Subprocess — This is a standard library use to process various system commands like to log off or to restart your PC.

- Json- The json module is used for storing and exchanging data.
- request- The request module is used to send all types of HTTP request. Its accepts URL as parameters and gives access to the given URL'S.
- wolfram alpha — Wolfram Alpha is an API which can compute expert-level answers using Wolfram's algorithms, knowledge base and AI technology. It is made possible by the Wolfram Language

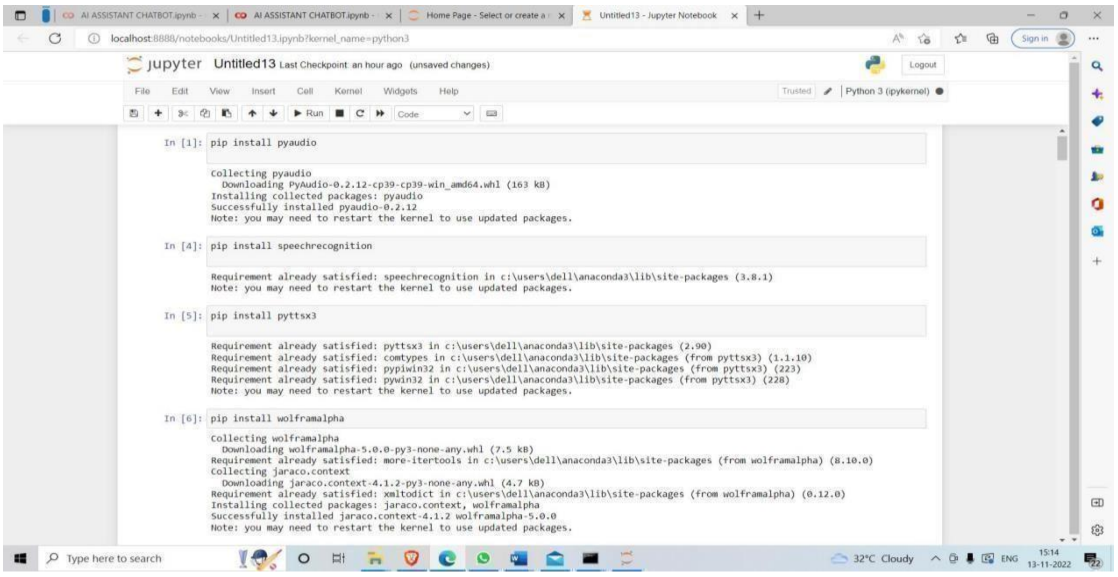
5. IMPLEMENTATION

```
from platform import uname
import time
import pyttsx3 # pip install pyttsx3
import speech_recognition as sr # pip install speechRecognition, pip install pipwin & pipwin
install pyaudio
import datetime # pip install datetime
import wikipedia # pip install wikipedia
import webbrowser
import os
import smtplib
import pyjokes # pip install pyjokes
import pyaudio

import subprocess
# initialises the pyttsx3 modules
# Microsoft Speech API (SAPI5) is the technology for voice recognition & synthesis,
provided by Microsoft
engine = pyttsx3.init('sapi5')
# it'll get the voice property frm the pyttsx3 module
voices = engine.getProperty('voices')
# it'll get list of voices & v can select 1 which v lyk
# voices[0].id -> Male voice
# voices[1].id -> Female voice
engine.setProperty('voice', voices[0].id)
# now vll set the voice speed
# by default it will b 200 words per min
engine.setProperty('rate', 190) # 190 words per min
ass_name = "AI"
def speak(audio):
   engine.say(audio) # convert the written text in2 speach
   engine.runAndWait() # pauses the program till the say function is done wyt speaking
def wishMe():
   speak("Welcome back!")
   hour = int(datetime.datetime.now().hour)
   if hour >= 2 and hour < 12:
      speak("Good Morning!")
   elif hour >= 12 and hour < 17:
      speak("Good Afternoon!")
    elif hour >= 17 and hour < 23:
      speak("Good Evening!")
   else:
      speak("Good Night!")
   speak("I'm your Assistant")
   speak(ass_name)
   speak("AI at your service")
def takeCommand():
   # It takes microphone input frm the user & returns string output
```

```
    r = sr.Recognizer() # v r initializing the Recognizer in the r variable
    with sr.Microphone() as source: # get input frm the user frm the microphone, it will our
source for our input
        print("Listening...")
        speak("Listening...")
        r.pause_threshold = 1 # it will wait for 1 sec before it strt 2 listen
        audio = r.listen(source) # microphone will listen 2 our audio, v hv passed source
variable in listen function
        try:
            print("Recognizing...")
            query = r.recognize_google(audio, language = 'en-in')
            print(f"User said: {query}\n")
        except Exception as e:
            # print(e)
            print("Could not understand audio...")
            speak("Could not understand audio...")
            print("Say that again please...")
            speak("Say that again please...")
            return "None"
        return query
def sendEmail(to, content):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    # enable low security in gmail
    server.login('your emai id', 'your email password')
    server.sendmail('your email id', to, content)
    server.close()
def extract_duration(input_str):
    # Extract numeric value from the input string
    words = input_str.split()
    for word in words:
        if word.isdigit():
            return word
    return None
def exit_program():
    print("Exiting the program...")
    # Perform any cleanup or finalization tasks if needed
    exit() # Terminate the program execution
if _name___== "_main_":
    wishMe()
    print('What should I call you ?')
    speak('What should I call you ?')
    uname = takeCommand()
    print('Welcome', uname)
    speak('Welcome')
    speak(uname)
    speak('Please tell me how may I help you')
while True:
    # if 1:
```

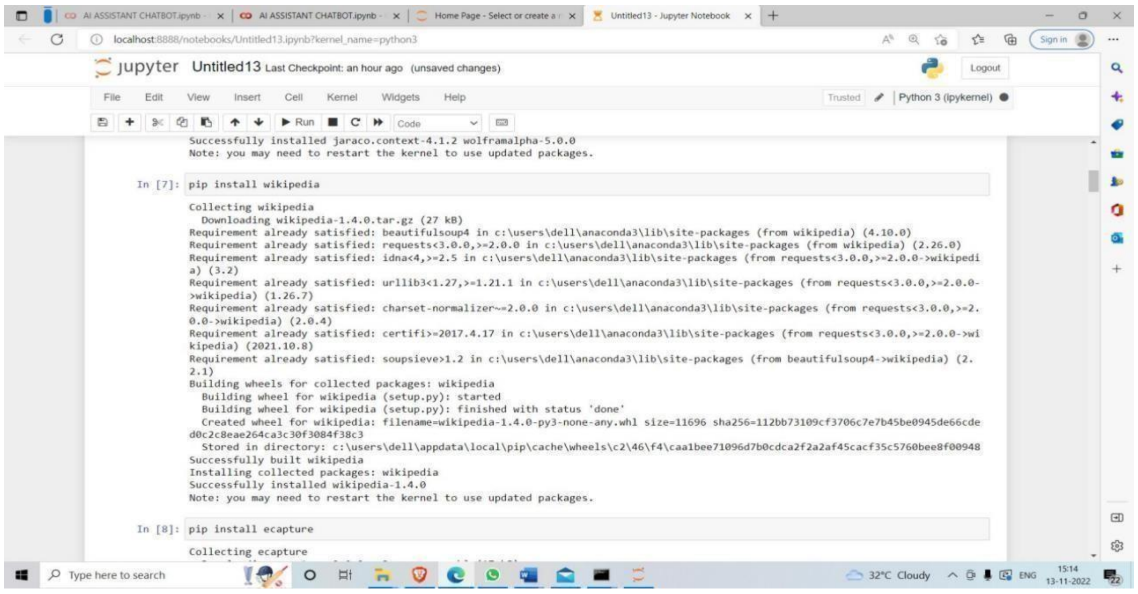
```
    query = takeCommand().lower()
    # logic for executing tasks based on query
    if 'wikipedia' in query:
        print('Searching Wikipedia...')
        speak('Searching Wikipedia...')
        results = wikipedia.summary(query.replace("wikipedia", ""), sentences = 2)
        speak("According to Wikipedia")
        print(results)
        speak(results)
    elif 'open youtube' in query:
        print('Here is your YouTube\n')
        speak('Here is your YouTube\n')
        webbrowser.open("youtube.com")
    elif 'open google' in query:
        print('Here is your Google\n')
        speak('Hereis your Google\n')
        webbrowser.open("google.com")
    elif 'open stackoverflow' in query:
        print('Here is your Stackoverflow\n')
        speak('Here is your Stackoverflow\n')
        webbrowser.open("stackoverflow.com")
    elif 'open github' in query:
        print('Here is your Github\n')
        speak('Here is your Github\n')
        webbrowser.open("github.com")
    elif 'open dev community' in query:
        print('Here is your Dev Community\n')
        speak('Here is your Dev Commmunity\n')
        webbrowser.open("dev.to")
    elif 'open protonmail' in query:
        print('Here is your Proton Mail\n')
        speak('Here is your Proton Mail\n')
        webbrowser.open("mail.protonmail.com")
    elif 'open brave' in query:
        print('Here is your Brave Browser\n')
        speak('Here is your Brave Browser\n')
        codePath=r"C:\\ProgramFiles\\BraveSoftware\\BraveBrowser\\Application\\r ave.exe"
        os.startfile(codePath)
    elif 'open firefox' in query:
        print('Here is your Firefox Browser\n')
        speak('Here is your Firefox Browser\n')
        codePath=r"C:\Program Files\Mozilla Firefox\firefox.exe" # ur firefox browser directory
        #os.startfile(codePath)
        webbrowser.open(codePath)
    elif 'open chrome' in query:
        print('Here is your Chrome Browser\n')
        speak('Here is your Chrome Browser\n')
        codePath=r"C:\ProgramFiles\Google\Chrome\Application\chrome.exe" # ur chrome
browser directory
        os.startfile(codePath)
        #webbrowser.open(codePath)
    elif 'open whatsapp' in query:
        print('Here is your WhatsApp\n')
        speak('Here is your WhatsApp\n')
        codePath = r"C:\Users\INDIA\Desktop\WhatsApp" # ur whatsapp directory
        os.startfile(codePath)
        #subprocess.Popen([codePath]
    elif 'open telegram' in query:
        print('Here is your Telegram\n')
        speak('Here is your Telegram\n')
        url='https://www.telegram.com/'
        webbrowser.open(url) # ur telegram directory
    elif 'open vs code' in query or 'visual studio code' in query:
        print('Here is your VS Code\n')
        speak('Here is your VS Code\n')
        codePath=r"C:\Users\INDIA\AppData\Local\Programs\Microsoft Code\Code.exe" # ur
vs code directory
        os.startfile(codePath)
    elif 'send a mail to' in query:
        try:
            speak("What should I say ?")
            content = takeCommand()
            speak("whom should i send")
            to = input()
            sendEmail(to, content)
            speak("Email has been sent!")
        except Exception as e:
            print(e)
            speak("Sorry my friend, I'm not able to send this email")
    elif 'the time' in query:
        strTime = datetime.datetime.now().strftime("%H:%M:%S")
        speak(f"The time now is {strTime}")
    elif 'how are you' in query:
        print("I'm fine, Thank you")
        speak("I'm fine, Thank you")
        print("How are you,", uname)
        speak("How are you,")
        speak(uname)
    elif 'fine' in query or 'good' in query:
        print("It's good to know that you are fine")
        speak("It's good to know that you are fine")
    elif 'who i am' in query or 'who am i' in query:
        print('If you talk then definitely your human and your name is', uname)
        speak('If you talk then definitely your human and your name is')
        speak(uname)
    elif 'change my name' in query:
        speak('What would you like me to call you ?')
        print('What would you like me to call you ?')
        uname = takeCommand()
        speak('Your name is changed to')
        speak(uname)
        print('Your name is changed to', uname)
    elif 'change your name' in query:
        print('What would you like to call me,', uname)
        speak('What would you like to call me,')
        speak(uname)
        ass_name = takeCommand()
        print('Thanks for naming me as', ass_name)
        speak('Thanks for naming me as')
        speak(ass_name)
    elif "what's your name" in query or 'what is your name' in query:
        print('My friends call me', ass_name)
        speak('My friends call me')
        speak(ass_name)
    elif 'what is my name' in queryor "what's my name" in query:
        print("Your name is ", uname)
        speak('Your name is')
        speak(uname)
    elif 'who made you' in query or 'who created you' in query:
        print('I have been created by a team')
        speak('I have been created by a team')
    elif 'exit' in query or 'quit' in query:
        print('Thanks', uname,'for giving me your time')
        speak('Thanks')
        speak(uname)
        speak('for giving me your time')
        exit()
    elif 'joke' in query:
        speak(pyjokes.get_joke())
    elif 'why you came to earth' in query or 'why did you come to earth' in query:
        print("Thanks to team, further It's a secret")
        speak("Thanks to team, further It's a secret")
    elif 'why have you been created' in query or 'reason for you' in query or 'reason for your
creation' in query:
        print('I was created as a project by a team')
        speak('I was created as a project by a team')
    elif 'where is' in query:
        location = query.replace('where is', '')
        print(uname, 'asked to locate', location)
        speak(uname)
        speak('asked to locate')
        speak(location)
        webbrowser.open("https://www.google.nl/maps/place/" + location + "")
    elif "write a note" in query:
        print('What should I write', uname)
        speak('What should I write')
        speak(uname)
        note = takeCommand()
        file = open('jarvis.txt', 'w')
        print(uname, 'Should I include date and time')
        speak(uname)
        speak("Should I include date and time")
        snfm = takeCommand()
        if 'yes' in snfm or 'sure' in snfm or 'ok' in snfm:
            strTime = datetime.datetime.now().strftime("%H:%M:%S")
            file.write(strTime)
            file.write(" :- ")
            file.write(note)
        else:
            file.write(note)
    elif "show the note" in query or 'open the note' in query:
        print("Showing Notes")
        speak("Showing Notes")
        file = open("jarvis.txt", "r")
        print(file.read())
        speak(file.read(6))
    elif "stop listening" in query or "don't listen" in query:
        speak("For how much time do you want to stop listening to commands?")
        sleep_time_input = takeCommand()
        if sleep_time_input is not None:
            duration = extract_duration(sleep_time_input)
            if duration is not None:
                sleep_time = int(duration)
                time.sleep(sleep_time)
                print(f"Stopped listening for {sleep_time} seconds.")
            else:
                print("Could not understand the duration. Please try again.")
        else:
            print("Could not understand audio. Please try again.")
            # most asked question frm google Assistant
    elif "will you be my gf" in query or "will you be my bf" in query:
        print("I'm not sure about, may be you should give me some time")
        speak("I'm not sure about, may be you should give me some time")
    elif "i love you" in query:
        print("It's hard to understand")
        speak("It's hard to understand")
    elif "know everything" in query:
        print("Because I am software")
        speak("Because I am software")
```

6. RESULT ANALYSIS



```
In [1]: pip install pyaudio

Collecting pyaudio
  Downloading PyAudio-0.2.12-cp39-cp39-win_amd64.whl (163 kB)
Installing collected packages: pyaudio
Successfully installed pyaudio-0.2.12
Note: you may need to restart the kernel to use updated packages.

In [4]: pip install speechrecognition

Requirement already satisfied: speechrecognition in c:\users\dell\anaconda3\lib\site-packages (3.8.1)
Note: you may need to restart the kernel to use updated packages.

In [5]: pip install pyttsx3

Requirement already satisfied: pyttsx3 in c:\users\dell\anaconda3\lib\site-packages (2.90)
Requirement already satisfied: comtypes in c:\users\dell\anaconda3\lib\site-packages (from pyttsx3) (1.1.10)
Requirement already satisfied: pypiwin32 in c:\users\dell\anaconda3\lib\site-packages (from pyttsx3) (223)
Requirement already satisfied: pywin32 in c:\users\dell\anaconda3\lib\site-packages (from pyttsx3) (228)
Note: you may need to restart the kernel to use updated packages.

In [6]: pip install wolframalpha

Collecting wolframalpha
  Downloading wolframalpha-5.0.0-py3-none-any.whl (7.5 kB)
Requirement already satisfied: more-itertools in c:\users\dell\anaconda3\lib\site-packages (from wolframalpha) (8.10.0)
Collecting jaraco.context
  Downloading jaraco.context-4.1.2-py3-none-any.whl (4.7 kB)
Requirement already satisfied: xmltodict in c:\users\dell\anaconda3\lib\site-packages (from wolframalpha) (0.12.0)
Installing collected packages: jaraco.context, wolframalpha
Successfully installed jaraco.context-4.1.2 wolframalpha-5.0.0
Note: you may need to restart the kernel to use updated packages.
```

Fig 6.1 Installation of pyaudio, speechrecognition, and pyttsx3



```
Successfully installed jaraco.context-4.1.2 wolframalpha-5.0.0
Note: you may need to restart the kernel to use updated packages.

In [7]: pip install wikipedia

Collecting wikipedia
  Downloading wikipedia-1.4.0.tar.gz (27 kB)
Requirement already satisfied: beautifulsoup4 in c:\users\dell\anaconda3\lib\site-packages (from wikipedia) (4.10.0)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in c:\users\dell\anaconda3\lib\site-packages (from wikipedia) (2.26.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dell\anaconda3\lib\site-packages (from requests<3.0.0,>=2.0.0->wikipedi
a) (3.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\dell\anaconda3\lib\site-packages (from requests<3.0.0,>=2.0.0-
>wikipedia) (1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\dell\anaconda3\lib\site-packages (from requests<3.0.0,>=2.
0.0->wikipedia) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dell\anaconda3\lib\site-packages (from requests<3.0.0,>=2.0.0->wi
kipedia) (2021.10.8)
Requirement already satisfied: soupsieve>1.2 in c:\users\dell\anaconda3\lib\site-packages (from beautifulsoup4->wikipedia) (2.
2.1)
Building wheels for collected packages: wikipedia
  Building wheel for wikipedia (setup.py): started
  Building wheel for wikipedia (setup.py): finished with status 'done'
  Created wheel for wikipedia: filename=wikipedia-1.4.0-py3-none-any.whl size=11696 sha256=112bb73109cf3706c7e7b45be0945de66cde
d0c2c8eae264ca3c30f3084f38c3
  Stored in directory: c:\users\dell\appdata\local\pip\cache\wheels\c2\46\f4\caa1bee71096d7b0cdca2f2a2af45cacf35c5760bee8f00948
Successfully built wikipedia
Installing collected packages: wikipedia
Successfully installed wikipedia-1.4.0
Note: you may need to restart the kernel to use updated packages.

In [8]: pip install ecapture

Collecting ecapture
```

Fig 6.2 Installation of Wikipedia

The above figures Fig 6.1 and Fig 6.2 shows the result of package installation i.e dependencies such as Speech Recognition audio input processing and NLTK for natural language understanding commonly installed to enhance functionality and accuracy.

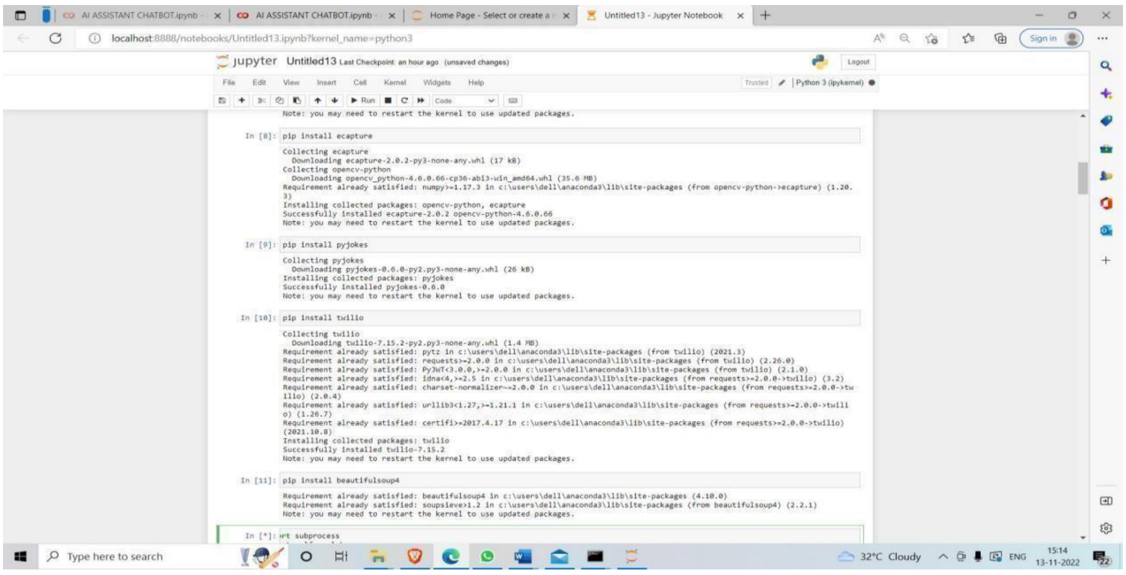

```
Note: you may need to restart the kernel to use updated packages.

In [8]: pip install ecapture

Collecting ecapture
  Downloading ecapture-2.0.2-py3-none-any.whl (17 kB)
Collecting opencv-python
  Downloading opencv_python-4.6.0.66-cp36-abi3-win_amd64.whl (35.6 MB)
Requirement already satisfied: numpy>=1.17.3 in c:\users\dell\anaconda3\lib\site-packages (from opencv-python->ecapture) (1.20.
3)
Installing collected packages: opencv-python, ecapture
Successfully installed ecapture-2.0.2 opencv-python-4.6.0.66
Note: you may need to restart the kernel to use updated packages.

In [9]: pip install pyjokes

Collecting pyjokes
  Downloading pyjokes-0.6.0-py2.py3-none-any.whl (26 kB)
Installing collected packages: pyjokes
Successfully installed pyjokes-0.6.0
Note: you may need to restart the kernel to use updated packages.

In [10]: pip install twilio

Collecting twilio
  Downloading twilio-7.15.2-py2.py3-none-any.whl (1.4 MB)
Requirement already satisfied: pytz in c:\users\dell\anaconda3\lib\site-packages (from twilio) (2021.3)
Requirement already satisfied: requests>=2.0.0 in c:\users\dell\anaconda3\lib\site-packages (from twilio) (2.26.0)
Requirement already satisfied: PyJWT<3.0.0,>=2.0.0 in c:\users\dell\anaconda3\lib\site-packages (from twilio) (2.1.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dell\anaconda3\lib\site-packages (from requests>=2.0.0->twilio) (3.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\dell\anaconda3\lib\site-packages (from requests>=2.0.0->tw
ilio) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\dell\anaconda3\lib\site-packages (from requests>=2.0.0->twili
o) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dell\anaconda3\lib\site-packages (from requests>=2.0.0->twilio)
(2021.10.8)
Installing collected packages: twilio
Successfully installed twilio-7.15.2
Note: you may need to restart the kernel to use updated packages.

In [11]: pip install beautifulsoup4

Requirement already satisfied: beautifulsoup4 in c:\users\dell\anaconda3\lib\site-packages (4.10.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\dell\anaconda3\lib\site-packages (from beautifulsoup4) (2.2.1)
Note: you may need to restart the kernel to use updated packages.

In [*]: rt subprocess
```

Fig 6.3 Installation of pyjokes and twilio

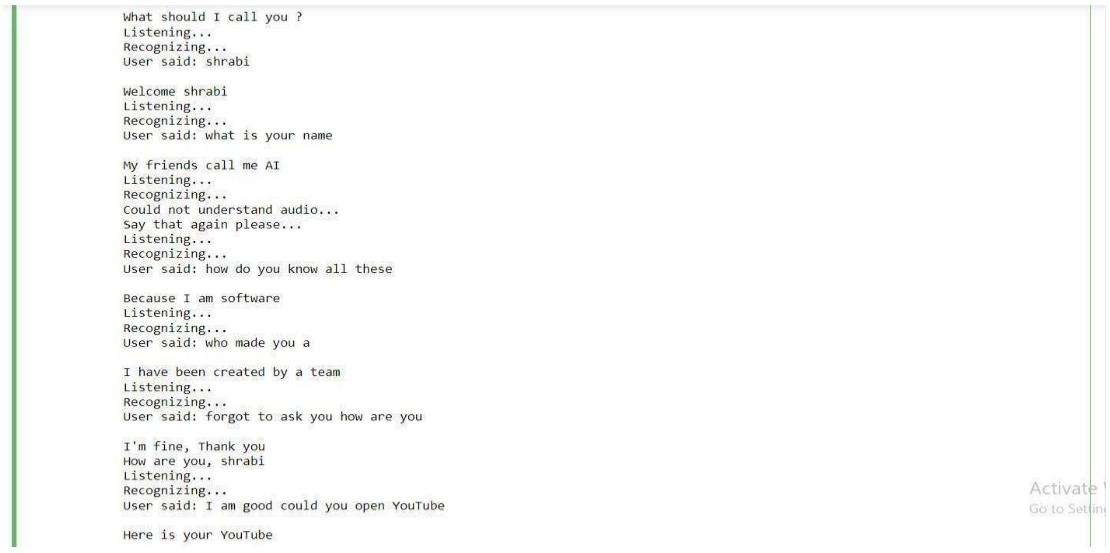

```
What should I call you ?
Listening...
Recognizing...
User said: shrabi

Welcome shrabi
Listening...
Recognizing...
User said: what is your name

My friends call me AI
Listening...
Recognizing...
Could not understand audio...
Say that again please...
Listening...
Recognizing...
User said: how do you know all these

Because I am software
Listening...
Recognizing...
User said: who made you a

I have been created by a team
Listening...
Recognizing...
User said: forgot to ask you how are you

I'm fine, Thank you
How are you, shrabi
Listening...
Recognizing...
User said: I am good could you open YouTube

Here is your YouTube
```

Fig 6.4 Output of voice assistant

Above Fig 6.3 represent the outputs of installing packages and Fig 6.4 represents the actual output of the voice assistant, where the commands or tasks are processed between the user and voice assistant.

7. Future Scope of AI in Voice Assistant

AI-driven voice assistants are evolving rapidly with advancements in deep learning, natural language processing (NLP), and human-computer interaction. Here are some key areas shaping the future of voice assistants:

1. Enhanced Speech Recognition & Multimodal Interaction

- **Improved Accuracy** – Advanced models like Whisper (OpenAI) and self-supervised learning will enhance ASR accuracy.
- **Multilingual & Code-Switching Support** – Seamless language switching within a conversation.
- **Multimodal AI** – Voice assistants will integrate with **vision (computer vision), gestures, and text** for richer interactions.

2. More Intelligent Natural Language Understanding (NLU)

- **Context-Aware Conversations** – AI will remember past interactions for smarter responses.
- **Personalized Assistants** – Customizable behavior based on user preferences.
- **Emotion & Sentiment Analysis** – Understanding user emotions to adjust responses accordingly.

3. Conversational AI & Dialogue Management Evolution

- **Few-shot & Zero-shot Learning** – AI assistants will require less training data to learn new tasks.
- **Reinforcement Learning (RLHF)** – Learning from human feedback to improve responses dynamically.

- **Autonomous AI Agents** – Future assistants will perform complex tasks independently

4. **More Natural & Expressive Text-to-Speech (TTS)**

- **Human-Like Speech Synthesis** – Using models like WaveNet, Tacotron, and FastSpeech for expressive speech.
- **Real-Time Voice Cloning** – Assistants can mimic any voice with just a few seconds of audio.
- **Emotionally Adaptive Speech** – TTS will adjust tone based on user emotions and context.

5. **Privacy, Security & Edge AI**

- **On-Device Processing** – Running AI models locally for **faster responses &** enhanced privacy (e.g., Apple's Siri Neural Engine).
- **Federated Learning** – Training AI models **without sharing user data** to the cloud.
- **Voice Biometrics** – Improved user authentication using unique voiceprints.

6. **Integration with IoT & Smart Devices**

.

- **AI in Healthcare** – Personalized virtual health assistants for diagnosis and mental wellness.
- **Enterprise Adoption** – AI-powered voice assistants for customer service, business automation, and virtual meetings

8. CONCLUSION

The final outcome of this project is an intelligent voice assistant. It combines natural language processing techniques with web programming to present an efficient digital personal assistant. It can give answers for some of the user questions. It can perform everyday tasks on the system based on the spoken command of the user. It can recognize the words, map the speech into text and decide what task to execute accordingly. The application can perform operations on the device such as open applications like chrome, whatsapp, youtube, firefox etc. It can also open websites, perform online searches and minimize all open windows on the computer.

Voice assistance powered by the artificial intelligence has significantly transformed the way humans interact with technology. By combining advanced algorithms in speech recognition, natural language understanding and dialogue management, Ai voice assistance can now respond to user commands in a more natural personalized and efficient manner.

While existing systems have laid a strong foundation they still face challenges such as limited contextual awareness privacy concerns and language diversity. However ongoing research and the development of more sophisticated AI models promise a future where voice assistants become even more intelligent ,adaptive and human - like.

As these systems continue to evolve they hold immense potential across domains such as healthcare , education ,customer service and smart house -making daily life more accessible ,inclusive and convenient for people around the world.

9.BIBLIOGRAPHY

[1] Cohen, J. (1960). "A Coefficient of Agreement for Nominal Scales." Educational and Psychological Measurement, 20(1), 37-46.

[2] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups." IEEE Signal Processing Magazine, 29(6), 82-97.

[3] Young, T., Hazarika, D., Poria, S., and Cambria, E. (2019). "Recent Trends in Deep Learning Based Natural Language Processing." IEEE Computational Intelligence Magazine, 14(4), 55-75.

[4] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). "Attention- Based Models for Speech Recognition." Advances in Neural Information Processing Systems (NeurIPS), 28, 577-585.

[5] Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. (2016). "Deep Reinforcement Learning for Dialogue Generation." Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 1, 1192-1202.

[6] Sarwar, A. F. M. (2020). "Voice Command Recognition System Using Deep Learning." International Journal of Computer Science and Information Security, 18(6), 30-34.

[7] Khosravi, M. R., and Aliakbarpour, H. (2018). "A Survey on Speech Recognition Systems." Journal of Intelligent Information Systems, 50(3), 385-403.

[8] Kim, Y., Park, Y., Oh, J., and Kim, H. (2021). "A Comprehensive Review of Voice- Based Virtual Personal Assistants." Electronics, 10(2), 164.

[9] Møller, M. F. (1993). "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning." Neural Networks, 6(4), 525-533.

[10] Weng, W., Yu, H., and Rui, Y. (2019). "Speech Command

Recognition Using Convolutional Neural Networks." Applied Sciences, 9(20), 432 Certainly! Here are some additional references for AI voice assistants. Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., et al. (2013). "Recent Advances in Deep Learning for Speech Research at Microsoft." Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 8604-8608.

[11] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). "Speech Recognition with Deep Recurrent Neural Networks." Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6645-6649.

[12] Kaldi Speech Recognition Toolkit. (2011). Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer. http://kaldi-asr.org/

[13] Ravanelli, M., and Bengio, Y. (2018). "Speaker Recognition from Raw Waveform with SincNet." Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, 201-205.

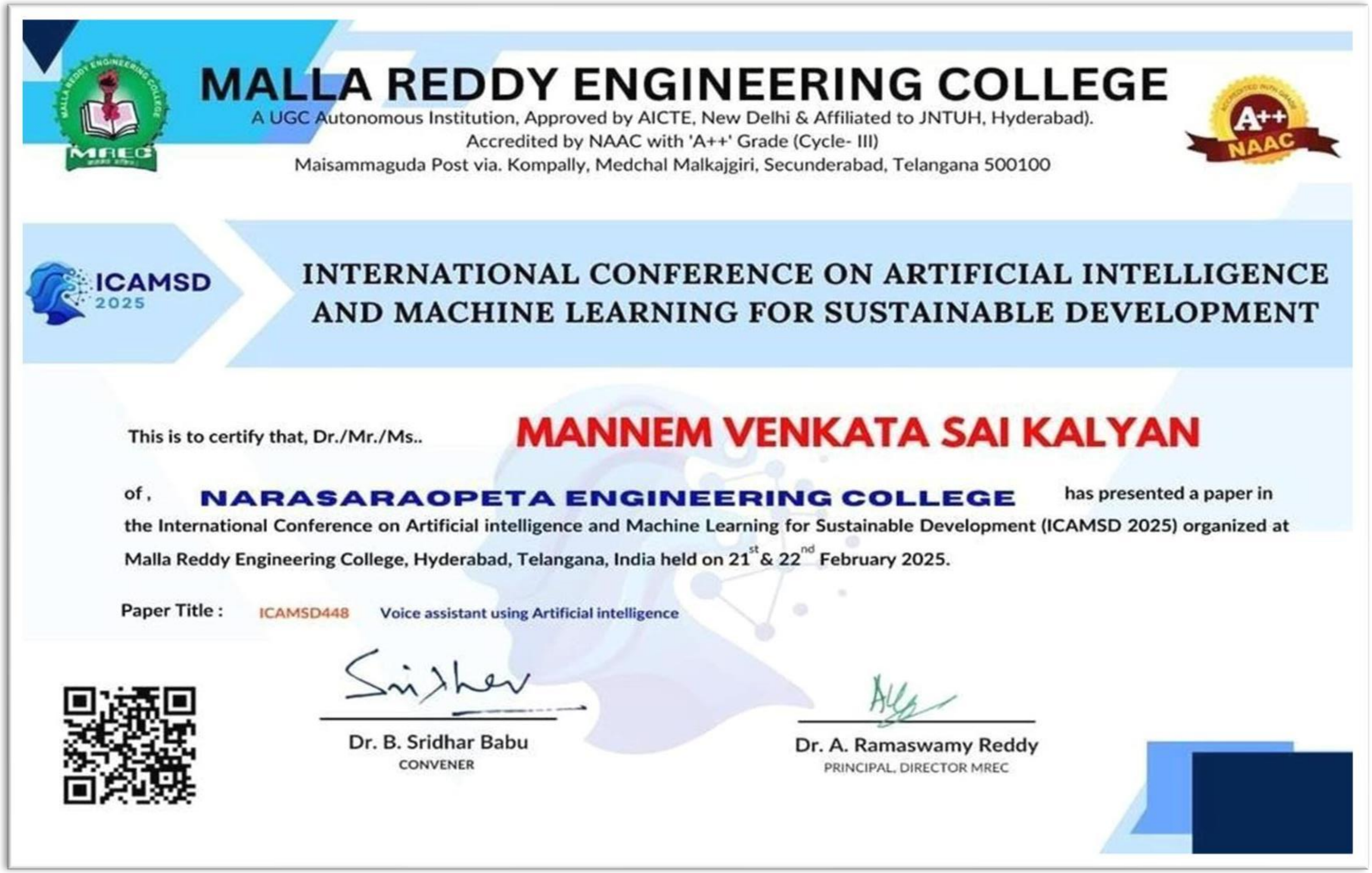
[14] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R. J., et al. (2018). "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions." Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, 3919-3928.

[15] Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., and Saurous, R. A. (2017). "Tacotron: Towards End-to-End Speech Synthesis." Proceedings of the 34th International Conference on Machine Learning (ICML), 2017, 3381- 3390.

[16] Wu, Y., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Johnson, J., Parmar, N., Vaswani, A., Shazeer, N., Young, C., et al. (2018). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." arXiv preprint arXiv:1609.08144.

[17] Zhang, Y., Chung, Y.-C., and Glass, J. (2017). "Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion." Proceedings of the Interspeech, 2017, 924- 928.

CONFERENCE CERTIFICATES



MALLA REDDY ENGINEERING COLLEGE

A UGC Autonomous Institution, Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad).
Accredited by NAAC with 'A++' Grade (Cycle- III)
Maisammaguda Post via. Kompally, Medchal Malkajgiri, Secunderabad, Telangana 500100

ICAMSD 2025

INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FOR SUSTAINABLE DEVELOPMENT

This is to certify that, Dr./Mr./Ms.. **MANNEM VENKATA SAI KALYAN**

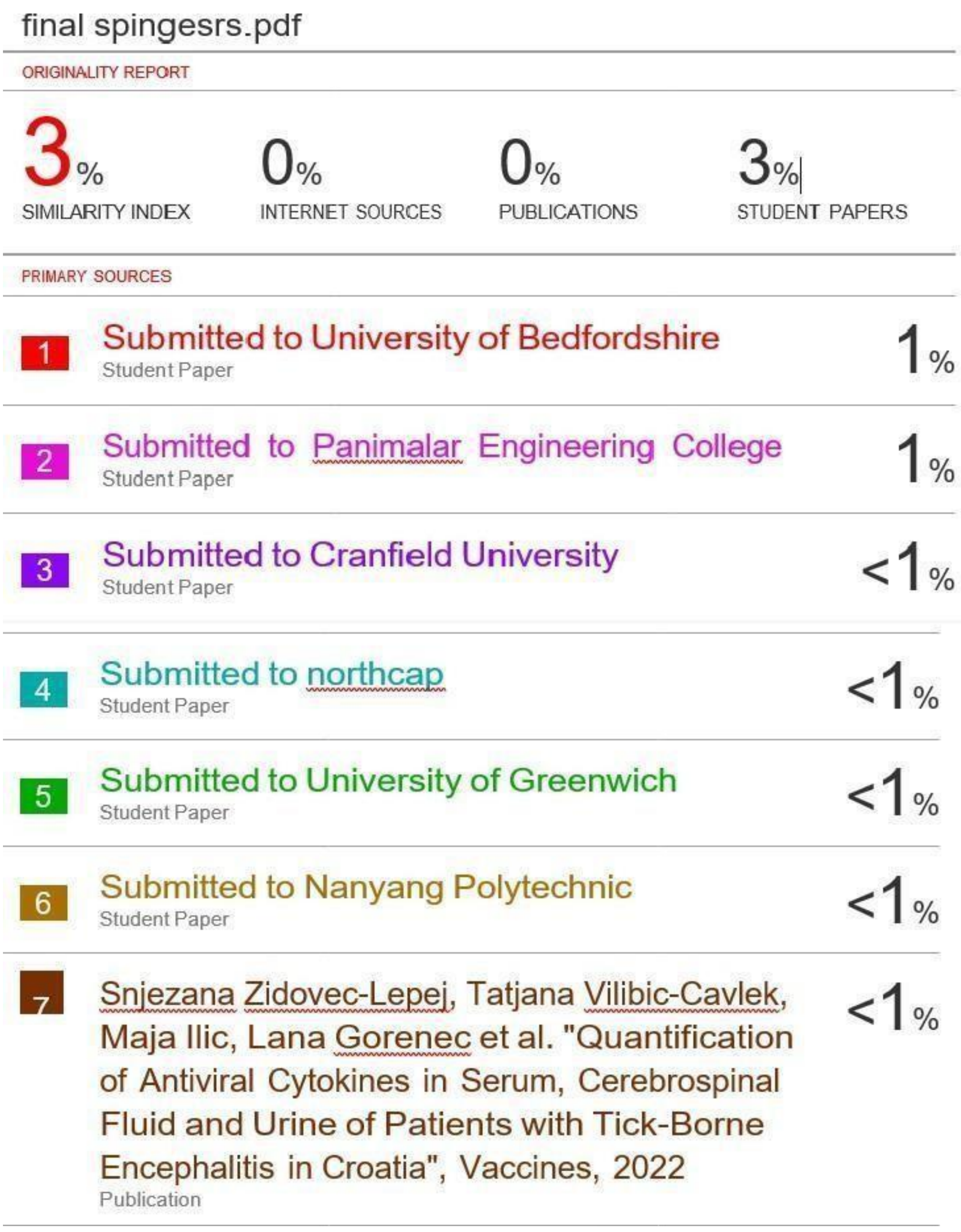
of , **NARASARAOPETA ENGINEERING COLLEGE** has presented a paper in the International Conference on Artificial intelligence and Machine Learning for Sustainable Development (ICAMSD 2025) organized at Malla Reddy Engineering College, Hyderabad, Telangana, India held on 21st & 22nd February 2025.

Paper Title : ICAMSD448 Voice assistant using Artificial intelligence

Dr. B. Sridhar Babu
CONVENER

Dr. A. Ramaswamy Reddy
PRINCIPAL, DIRECTOR MREC

PLAGIARISM REPORT



final spingesrs.pdf

ORIGINALITY REPORT

3%	0%	0%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Bedfordshire Student Paper	1%
2	Submitted to Panimalar Engineering College Student Paper	1%
3	Submitted to Cranfield University Student Paper	<1%
4	Submitted to northcap Student Paper	<1%
5	Submitted to University of Greenwich Student Paper	<1%
6	Submitted to Nanyang Polytechnic Student Paper	<1%
7	Snjezana Zidovec-Lepej, Tatjana Vilibic-Cavlek, Maja Ilic, Lana Gorenec et al. "Quantification of Antiviral Cytokines in Serum, Cerebrospinal Fluid and Urine of Patients with Tick-Borne Encephalitis in Croatia", Vaccines, 2022 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

CONFERENCE PAPER

Voice Assistant using Artificial Intelligence

Manohar Tinnavali [1*], Venkata Sai Kalyan Mannem [2], Rakesh Bhavirisetty[3], Ajay Jampani[4], Dr. Jhansi Vazram Bolla[5]

[1]Assistant Professor, Department of Information Technology, Narasaraopeta Engineering College, Narasaraopeta, Palnadu District, 522601, Andhra Pradesh, India.

[2,3,4] Student, Department of Information Technology, Narasaraopeta Engineering College, Narasaraopeta, Palnadu District, 522601, Andhra Pradesh, India.

[5]Professor & Head, Department of Information Technology, Narasaraopeta Engineering College, Narasaraopeta, Palnadu District, 522601, Andhra Pradesh, India.

**Corresponding Author(s) Email: manucse1850@gmail.com*

Contributing Authors: mannemvenkatasaikalyan@gmail.com, rakeshbhavirisetty@gmail.com, ajayjampani19@gmail.com, jhansi.bolla@gmail.com

Abstract: Voice assistants represent a transformation in human interaction commands. This project aims at designing and developing a statically voice - enabled personal assistant . It utilizes Python with strong emphasis on accessibility to disable people . Unlike the traditional input methods, the proposed system uses speech recognition and natural language processing to convert speech into text and then analyze the intent for file manger , web browsing, sending text messages, and even online orders. The assistant functions with ongoing listening, turning on only when the wake-up word is sensed. The core functionality includes automatic tasking, web scraping, and ease of interface with applications like YouTube, Wikipedia, or messaging. The modular construction of the system, run by the Python libraries - pyttsx3, Speech Recognition, and PyAudio- ensures its extensibility and ability to accommodate the new functionalities without disrupting previous functionality. This project would be to showcase how cutting-edge technologies such as AI, NLP, and TTS improve usability and accessibility. Because of its independence in performing tasks alone, users are eventually freed from their dependency on physical inputs, not to mention empowered, with the simplification of everyday tasks. From the management of personal tasks to the purchase of online items, for instance, makes it a valuable tool in daily life. The modular and open-source nature of the system underlines its scope for further innovation, opening it to future upgrades in voice -based intelligent systems.

Keywords: Voice assistants, text-to-speech, speech recognition, chatbots, and static voice assistants.

INTRODUCTION

Digital devices that use speech synthesis, natural language processing algorithms, and human voice input to react to spoken instructions are known as voice assistants [1]. These systems are designed to respond to user requests, sometimes known as motives, by delivering pertinent information or carrying out specific actions [2]. In order to provide accurate responses, voice assistants identify keywords from input by users while removing background noise [2]. Some voice assistants, like Amazon Alexa-enabled devices, are made especially for a single application, while others are software-based and work with a variety of devices [4]. These days, virtual assistants are built into commonplace gadgets like computers, cellphones, and smart speakers [4].

LITERATURE SURVEY

This innovation has been driven by the latest breakthrough of artificial intelligence and natural language processing, giving tremendous momentum to voice assistants. An innovation called the Flat Direct Model (FDM) was introduced by Patrick Nguyen et al. This removed traditional markov models, along with simplifying feature definitions, which helped improve speech recognition accuracy and reduced the rate of errors within a sentence-this provided a strong foundation for enhancing voice recognitions in assistants. Their contribution demonstrates the use of a microcontroller with Wi-Fi and sensor integration that helps to let assistants see devices, control devices such as lights, fans, and so on. On mobile platforms, intelligent voice assistants by Sutar Shekhar et al., with scope for predicting behavior and personal recommendation have been developed. This helps develop voice assistants adaptable to users' general preferences.

Building up on NLP is one of the further things pursued by Rishabh Shah et al., because such integration helped in describing commands in varied languages hence creating more access. Moreover, Luis Javier Rodrıguez-Fuentes et al conducted efficient use of spoken term detection. There the advanced techniques of processing have helped in using large quantities of voice data. Finally, Tanvee Gawandetal. developed a flexible and easy-to-develop modular, offline-capable voice assistant by combining Python with Artificial Intelligence Markup Language (AIML) and Google Text-to-Speech (gTTS).

PROPOSED SYSTEM

System Design System Design comprised of the following stages:
Input Phase The microphone captures speech patterns. Speech patterns are captured bya Microphone.

Audio Data Recognition

Recognizing audio and it is converting into a text data by utilizing Natural Language Processing (NLP)on the captured audio. Cross Validation Text converted from audio is cross-validated against predefined commands. Output Generation Based on the input and validation, the desired output is created.

Phase 1: Gathering of Information

Speech patterns that are imported from the microphone as audio data.

Phase 2: Audio Processing

The captured audio data is handled and converted into text by applying NLP techniques to it.

Phase 3: Processing Data

Text data management is done through Python scripts to finalize the required output procedures.

Phase 4: Final Output Presentation

The final output might either be displayed as text on a screen or be converted back into speech with the help of Text-to-Speech (TTS) technology.

Features Tasks that can be executed by the designed system will be: Continuous Listening This allows the system to continuously listen without the user's operations and act based on certain user-initiated specific commands. Surfing the Web The system will search based on user-input parameters given orally by the user. The desired results are given back as output that simultaneously leads to screen.

SYSTEM ARCHITECTURE

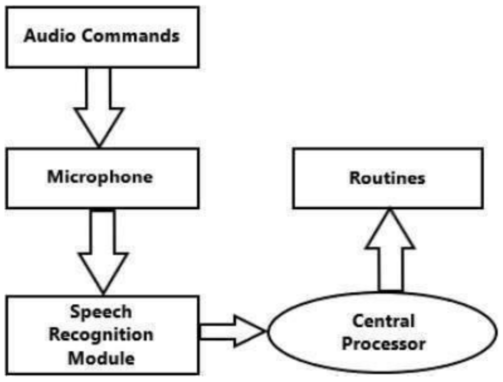


Fig1 This figure is able to symbolize and illustrate the flow of a Voice-Activated assistant

Basic workflow

The voice assistant workflow begins with speech recognition, which translates the user's speech input into text. The processor receives this textual data, where the system analyzes the nature of the command and invokes the corresponding script for execution. However, with ambient noise, the efficiency of the system is compromised. For example, even if the data have been inputted for hours, it may not correctly identify or process a voice command if the device interferes with dog barking or helicopter noise. The reason is that noise filtering methods should be carried out to ensure accurate recognition and processing.

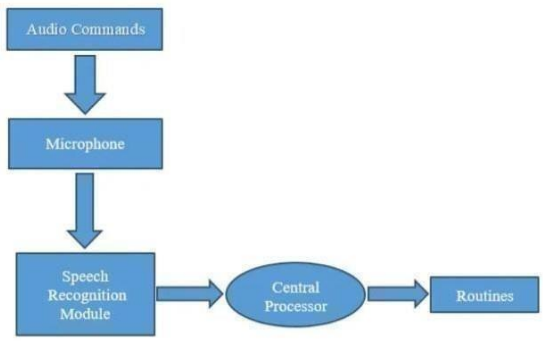


Fig2 This figure illustrates the fundamental workflow of a voice assistant

Detailed Workflow

Our interactions with technology have evolved as a result of voice assistants like Siri, Google Voice, and Bixby. They are everywhere today, in our pockets through smartphones or in smart speakers like the Amazon Echo and Google Home. An NPR report indicates that nearly six Americans already owns a smart speaker for home use, and their adoption rate is comparable to the explosive growth of smartphones a decade ago.

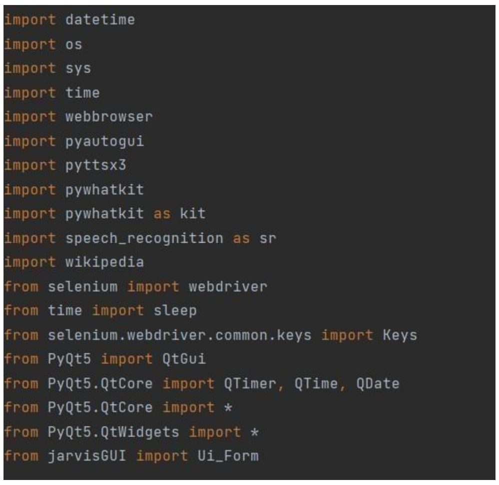
These devices have transformed daily activities, from setting reminders to controlling smart home systems, making life more convenient and connected. Despite this progress, voice assistants are still less widely adopted in workplaces. One significant deterrent is the shift toward open office environments, where privacy is limited and speaking to a virtual assistant aloud could disrupt colleagues. Fear of being perceived as inconsiderate or unprofessional inhibits the use of such tools in professional settings.
Workplace use cases for voice assistants require more personalized functionalities, including managing team schedules, summarizing meetings, or automating complex workflows. If there are no proper features that deal with professional requirements while keeping privacy intact, then voice technology will not thrive in workplaces.

Voice Input Acquisition

The assistant receives voice input from the user in the first module. The mod- ule captures the analog voice of the user through a microphone and prepares it for processing. Input Analysis and Intent Mapping. The captured input is analyzed using natural language processing techniques. The analog voice input is converted into digital text that is mapped to the respective intent and function. This step ensures that the assistant accurately understands the user's command and identifies the corresponding action.

Output Generation and Response

This module refers to the delivery of the result to the user. The assistant processes the mapped intent and performs the appropriate function, and the output is received in a voice format accompanied by any required visual feedback on the screen.



```
import datetime
import os
import sys
import time
import webbrowser
import pyautogui
import pyttsx3
import pywhatkit
import pywhatkit as kit
import speech_recognition as sr
import wikipedia
from selenium import webdriver
from time import sleep
from selenium.webdriver.common.keys import Keys
from PyQt5 import QtGui
from PyQt5.QtCore import QTimer, QTime, QDate
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *
from jarvisGUI import Ui_Form
```

Fig 3 Imported modules

METHODOLOGIES

Methodology To make the voice assistant able to process speech, we use the system's voice functionality through SAPI5 and the pyttsx3 library. Text-to-speech conversion is made possible via the Python package pyttsx3. It is compatible with both Python 2 and Python 3, and unlike other libraries, it operates offline. A programmatic interface called the Speech Application Programming Interface was created by Microsoft to enable speech synthesis and recognition in Windows programs. Encapsulating all of the assistant's skills is its primary function. system has the following features built into its design:

1. Voice Command Listening The assistant is always listening for user input. The listening time can be set according to the user's preference.
2. Handling Misunderstood Commands If the assistant does not understand a command, it will ask the user to repeat the input until the command is understood.
3. Voice Customization The assistant can be set to use either a male or female voice, as preferred by the user. Features It is capable of nearly everything that the current assistant can do, such as: A weather report

Sending and receiving emails Look it up on Wikipedia Open and close programs The time check for now Take and display notes. Open Google and YouTube. Manage each application's window by opening or closing it.

4. Methodologies that enable the voice assistant to be interacting with users very effectively while giving great levels of customization and wide usage of features.

Voice Recognition Module

To activate voice recognition in the voice assistant app, the assistant should be trained to recognize the voice of the user. Use the following command in the terminal to install the module needed.

Date and Time Module

The Date Time package is a built-in module in Python that allows you to display the current date and time.

A OS

Python's OS module is used to communicate with the operating system. One of Python's standard utility modules, it provides access to functionality that depends on the operating system.

The Python Backend

"The Python backend further processes the output from the voice recognition module. This is in determining whether it's an API call or whether the context should be extracted from the speech command. Then the output goes back to the Python backend for processing and forwarding to the appropriate user."

Pyaudio

Pyaudio is a collection of Python bindings for PortAudio and a cross- platform C++ library that connects to audio drivers to allow Python applications to use audio input and output.

PyQt5

A complete collection of Python bindings for Qt is called PyQt5.With the help of PyQt5, which is implemented as more than 35 extension modules, Python can be used as a C++ substitute on any systems that accept it, including iOS and Android.

Text to Speech Module

The "Text-to-Speech (TTS) module lets computers read and hear text. A TTS engine translates written text into phonemic form and then renders the phonemic forms into waveforms that can be sent as audio. Many third-party vendors supply TTS engines with support for various languages, dialects, and specialized vocabularies.

Speech to text conversion

" To turn speech input into written output, speech recognition is utilized. It makes the spoken voice easier for the computer to process and comprehend by decoding it and turning it into text".

Content Extraction

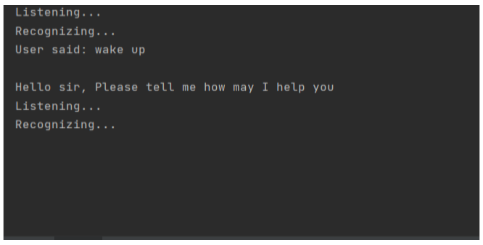
The process of automatically extracting structured data from machine-readable, unstructured or semi-structured documents is known as context extraction, or CE. In general, natural language processing (NLP) is used to process texts written in human languages. One could consider recent developments in multimodal document processing, such as content extraction from audio, video, and photos and automatic annotation, to be the result of context extraction approaches.

Textual Output

After deciphering the voice command and carrying out the action, it outputs the voice command as text in the terminal.

EXECUTION

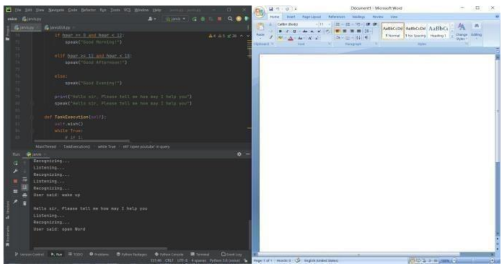
"The assistant waits until the user starts inputting during boot. On the user making a voice command,



```
Listening...
Recognizing...
User said: wake up

Hello sir, Please tell me how may I help you
Listening...
Recognizing...
```

the assistant captures the input and looks up for keywords it deems pertinent. When there is a corresponding keyword, it performs the said task and conveys the result in both voice and as a text in the terminal window. If there are no relevant keywords, the assistant will ask for a valid command from the user. Every feature has its vital role in achieving the seamless function of the system. Either wait for the user to click on ’start’ or ’quit’ to star.”

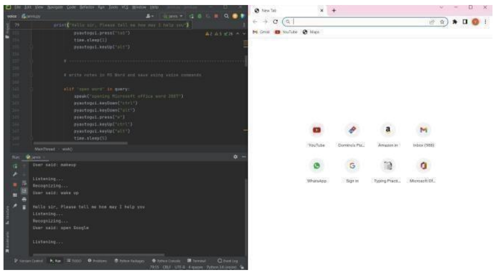
Fig 4 The wake up command



```
        if hour >= 0 and hour < 12:
            speak("Good Morning!")

        elif hour >= 12 and hour < 18:
            speak("Good Afternoon!")

        else:
            speak("Good Evening!")

        print("Hello sir, Please tell me how may I help you")
        speak("Hello sir, Please tell me how may I help you")

    def TaskExecution(self):
        self.wish()
        while True:
            # if 1:
```

```
Recognizing...
Listening...
Recognizing...
Listening...
Recognizing...
User said: wake up

Hello sir, Please tell me how may I help you
Listening...
Recognizing...
User said: open Word
```

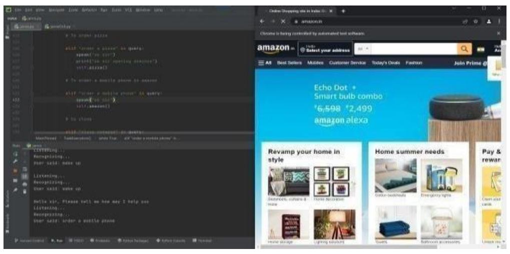
Fig. 5 A Voice command to Opening the Microsoft Office Word



```
        print("Hello sir, Please tell me how may I help you")
                pyautogui.press("tab")
                time.sleep(1)
                pyautogui.keyUp("alt")

            # ----------------------------------------------------------

            # write notes in MS Word and save using voice commands

            elif "open word" in query:
                speak("opening Microsoft office word 2007")
                pyautogui.keyDown("ctrl")
                pyautogui.keyDown("alt")
                pyautogui.press("w")
                pyautogui.keyUp("ctrl")
                pyautogui.keyUp("alt")
                time.sleep(5)
```

```
User said: makeup

Listening...
Recognizing...
User said: wake up

Hello sir, Please tell me how may I help you
Listening...
Recognizing...
User said: open Google

Listening...
```

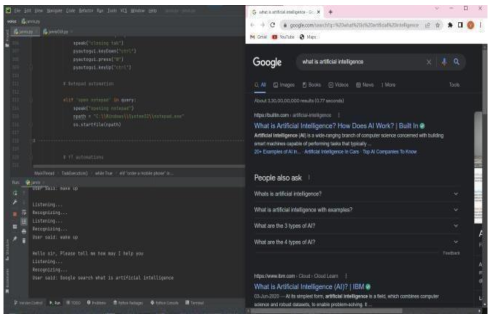
Fig. 6 A Voice Command to open the Google Chrome



```
            # To order pizza

            elif "order a pizza" in query:
                speak("ok sir")
                print("ok sir opening dominos")
                self.pizza()

            # To order a mobile phone in amazon

            elif "order a mobile phone" in query:
                speak("ok sir")
                self.amazon()

            # To close
```

```
Listening...
Recognizing...
User said: wake up

Listening...
Recognizing...
User said: wake up

Hello sir, Please tell me how may I help you
Listening...
Recognizing...
User said: order a mobile phone
```

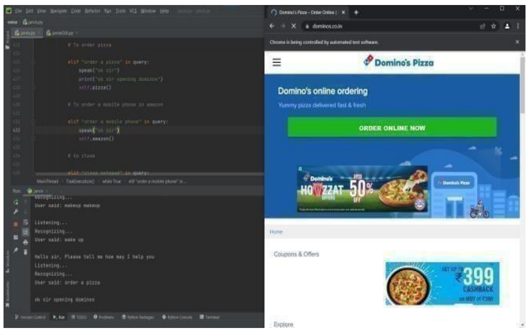
Fig. 7 A Voice command to order a mobile in amazon



```
            speak("closing tab")
            pyautogui.keyDown("ctrl")
            pyautogui.press("W")
            pyautogui.keyUp("ctrl")

        # Notepad automation

        elif "open notepad" in query:
            speak("opening notepad")
            npath = "C:\\Windows\\System32\\notepad.exe"
            os.startfile(npath)

        # YT automations
```

```
User said: wake up

Listening...
Recognizing...
Listening...
Recognizing...
User said: wake up

Hello sir, Please tell me how may I help you
Listening...
Recognizing...
User said: Google search what is artificial intelligence
```

Fig. 8 A voice command used to gather the Information From Google



```
        # To order pizza

        elif "order a pizza" in query:
            speak("ok sir")
            print("ok sir opening dominos")
            self.pizza()

        # To order a mobile phone in amazon

        elif "order a mobile phone" in query:
            speak("ok sir")
            self.amazon()

        # to close

        elif "close notepad" in query:
```

```
Recognizing...
User said: makeup makeup

Listening...
Recognizing...
User said: wake up

Hello sir, Please tell me how may I help you
Listening...
Recognizing...
User said: order a pizza

ok sir opening dominos
```

Fig. 9 A Voice Command to Order a Pizza in Dominos

CONCLUSION

This paper takes into account providing an overview regarding the model and develop- ment of static and voice-enabled personal assistants to be used by PCs with regard to the language, Python programming. In this present generation the fast-paced lifestyles, and this kind of voice assistant is designed to perform the task to be more helpful in saving time rather than traditional methodologies for support needed for differently abled individuals and also performs as instructed by users. This smart device can forward a message to the user’s phones, control YouTube tasks, and even retrieve information from the Wikipedia, Google just through one command via voice. This voice assistant can perform several tasks by giving simple commands, hence they are making everyday web search for most people quite simple and accessible. This objective of this project is used to develop the assistant for further to become a full - fledged server assistant that can potentially functionally replace general server administration tasks.Open-source software modules compose the system, along with the support of PyCharm’s open community, to be easily and constantly updated.The system’s modularity benefits from flexibility while adding new features not affecting any existing system features. This design allows the assistant to evolve according to new innovations, which means it would be more efficient and applicable with time.

DISCUSSION

This project has successfully demonstrated how voice-enabled personal assistants can make complex everyday tasks quite easy and can be managed with speech-based com- munication. Merging together solutions like Speech Recognition, Text-to-Speech, and Natural Language Processing using Python, it can do so much: manage files, surf the Internet, send messages, or even order online. It is particularly useful to people with disabilities since it provides an interface that is voice-enabled so that they can interact with technologies easily and successfully. This system is highly remarkable for its modularity; it allows easy up-gradation and addition of new functionalities without disturbing the existing ones. Open-source libraries ensure that the system remains open and cost-effective, hence turning out to be a very practical solution for so many applications. Tasks such as retrieving information from Wikipedia, searching the web, and interacting with system files make the system-versatile and efficient. While the current version would suffice for the defined tasks, much more has yet to be added. Future versions might emerge with machine learning providing the response that suits personal conditions, support for various languages that can reach out to a much larger audience, and heightened contextual understanding to solve a number of more complex questions that may be raised. All of the foregoing would support moving an assistant from a side tool to one which, in practical life, turns out really smart and indispensable.

RESULT

The static voice-enabled personal assistant was developed and implemented. The results of the development and implementation were successful as the system was able to perform a variety of tasks efficiently. The results of testing and evaluation are summarized below:

Task Automation

The assistant was able to execute predefined commands like opening applications, managing files, retrieving information from Wikipedia, and performing web searches.
The tasks of sending text messages and ordering online, such as ordering pizza or buying items, were performed correctly and timely using voice commands.

Speech Recognition Accuracy

The assistant was highly accurate in recognizing user commands in a noise- free environment. - It was capable of distinguishing between legitimate commands and irrelevant ambient noise. The recognition accuracy in lab conditions was about 90

Text-to-Speech Feedback

The system could provide vocal responses to questions and, using the pyttsx3 library, create understandable, natural speech. There were simultaneous text-based outputs to accommodate users who learn visually and can read instead.

Ease of Use

The assistant responded to wake-up prompts promptly and executed the user’s inputs with no discernible delay in time. - The graphical user interface of PyQt5 has given enhanced user interaction and made the system intuitive and user-friendly.

Flexibility

The assistant was flexible in doing the task. It can do the task based on a wide range of areas, such as gathering data, controlling the system automatically, and web browsing. The modular architecture design ensured that more features would be added using some aspects of testing.
Accessibility FeaturesThe assistant was very effective for different kinds of abilities. The ability will give hands-free access to tasks that require manual input normally.

Observed Limitations

*The assistant was unable to hear commands in noisy situations or with accents and dialects not represented in the training data. * Complex, multi-steps instructions required further elaboration on the logical process to further refine the precision. On the whole, the results demonstrate that the system functioned correctly and can be used in practice to make everyday tasks easier. At the same time, results of these exper- iments indicate aspects where the system requires further development, for instance, increasing speech robustness and refining the natural language understanding.

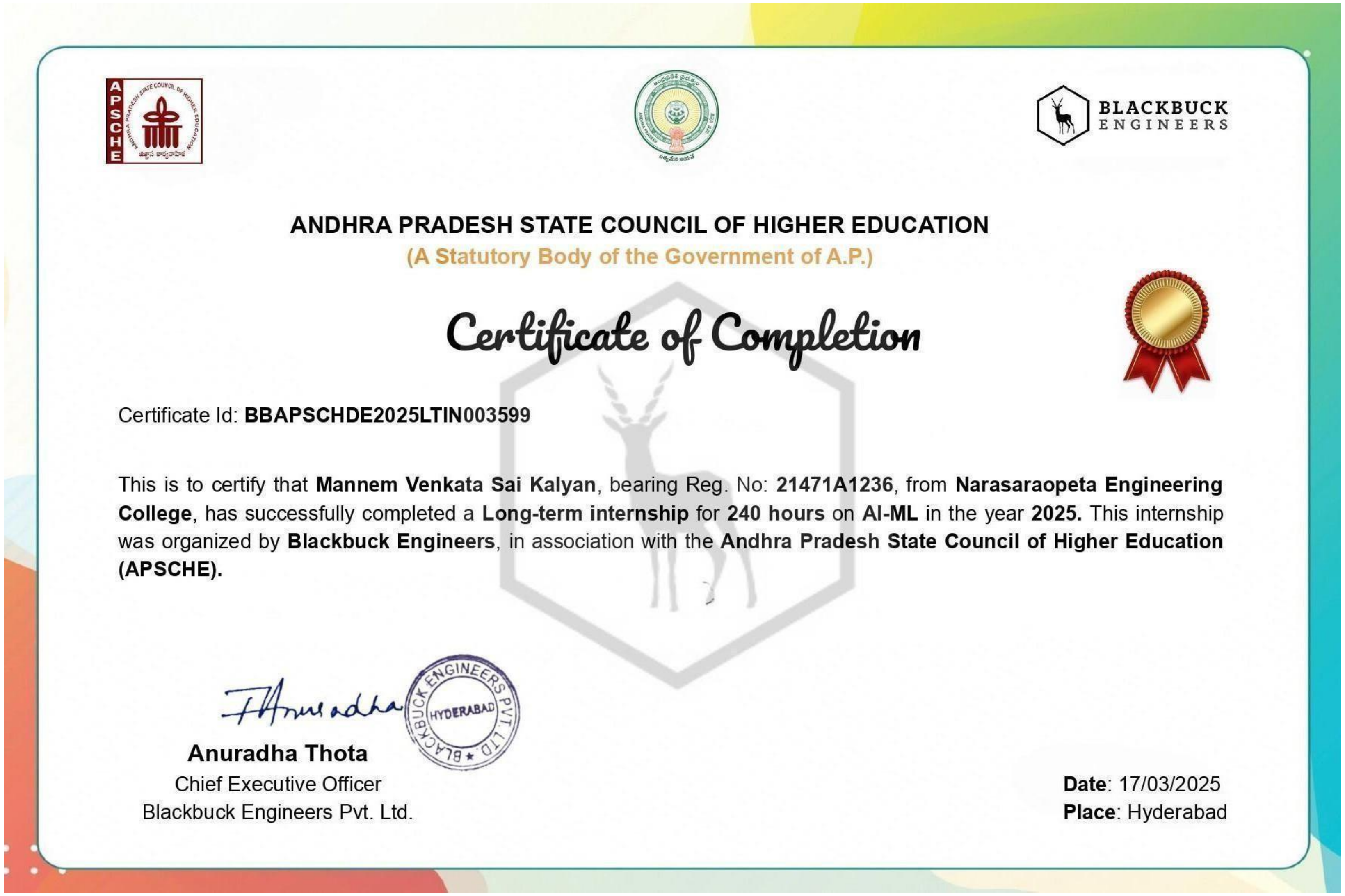
REFRENCES

1. Shaughnessy, IEEE, “Interacting with Computers by Voice: Automatic Speech Recognition and Synthesis” proceedings of the IEEE, vol. 91, no. 9, september 2022.

2. Patrick Nguyen, Georg Heigold, Geoffrey Zweig, “Speech Recognition with Flat Direct Models”, IEEE Journal of Selected Topics in Signal Processing, 2021.
3. Mackworth (2019-2020), Python code for voice assistant: Foundations of Computational Agents- David L. Poole and Alan K. Mackworth.
4. Nil Goksel, CanbekMehmet ,EminMutlu, “On the track of Artificial Intelligence: Learning with Intelligent Personal Assistant”, proceedings of International Journal of Human Sciences

5. Keerthana S, Meghana H, Priyanka K, Sahana V. Rao, Ashwini B “Smart Home Using Internet of Things ”, proceedings of Perspectives in Communication , Embedded -systems and signal processing.

INTERSHIP CERTIFICATES



BLACKBUCK ENGINEERS

ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

(A Statutory Body of the Government of A.P.)

Certificate of Completion

Certificate Id: **BBAPSCHDE2025LTIN003599**

This is to certify that **Mannem Venkata Sai Kalyan**, bearing Reg. No: **21471A1236**, from **Narasaraopeta Engineering College**, has successfully completed a **Long-term internship** for **240 hours** on **AI-ML** in the year **2025.** This internship was organized by **Blackbuck Engineers**, in association with the **Andhra Pradesh State Council of Higher Education (APSCHE).**

[illegible]

BLACKBUCK ENGINEERS PVT. LTD. ★ HYDERABAD

Anuradha Thota
Chief Executive Officer
Blackbuck Engineers Pvt. Ltd.

Date: 17/03/2025
Place: Hyderabad