Interactive Voice Assistant using Machine Learning Techniques

A PROJECT REPORT

Submitted by,

Rohit Jayaraj – 20201CDV0025
Arun Philip Sigeon – 20201CDV0018
Chaitra N – 20201CDV0019
Mohammed Rinshad – 20201CDV0010
D. Sai Shashank – 20201CDV0013

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (DEVOPS).

Under the guidance of,

Ms. B. Parkavi
Assistant Professor - SoCSE

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2024

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report **"Interactive Voice Assistant using Machine Learning Techniques"** being submitted by "Rohit Jayaraj – 20201CDV0025, D. Sai Shashank – 20201CDV0013, Chaitra N – 20201CDV0019, Arun Philip Sigeon – 20201CDV0018, Mohammed Rinshad – 20201CDV0010" in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering (DevOps) is a bonafide work carried out under my supervision.

Ms. B. Parkavi
Assistant Professor
School of CSE&IS
Presidency University

Dr. Senthil Kumar
Professor & HoD
School of CSE&IS
Presidency University

Dr. C. KALAIARASAN
Associate Dean
School of CSE&IS
Presidency University

Dr. SHAKKEERA L
Associate Dean
School of CSE&IS
Presidency University

Dr. Md SAMEERUDDIN KHAN
Dean
School of CSE&IS
Presidency University

Name of the Examiner and Signature:
1)
2)

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Interactive Voice Assistant using Machine Learning** in partial fulfilment for the award of Degree of **Bachelor of Technology** in **Computer Science and Engineering (DevOps)**, is a record of our own investigations carried under the guidance of **Ms. B. Parkavi, Assistant Professor, School of Computer Science Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Sl. No.	Name	Roll Number	Signature
1	D. Sai Shashank	20201CDV0013	
2	Rohit Jayaraj	20201CDV0025	
3	Chaitra N	20201CDV0019	
4	Arun Philip Sigeon	20201CDV0018	
5	Mohammed Rinshad	20201CDV0010	

ABSTRACT

This research presents an innovative approach to enhance the capabilities of interactive voice assistants through the integration of advanced machine learning techniques. The proposed system leverages state-of-the-art natural language processing (NLP) models, speech recognition algorithms, and context-aware machine learning to create a more intuitive and personalized user experience. The incorporation of deep learning models enables the system to continuously improve its understanding of diverse linguistic nuances, ensuring effective communication with users across various domains. Voice assistants like Siri, Alexa, and Google Assistant have become household names, revolutionizing how we interact with technology. Behind these interfaces lies a complex web of machine learning techniques. This report provides a concise overview of how these Machine Learning techniques power voice assistants. Voice assistants can be used for a multitude of purposes, such as retrieving data from the internet, including news stories, Wikipedia pages, and weather predictions, as well as providing answers to a wide range of questions. They can also be used to send emails and messages, as well as to complete chores like making to-do lists, sending messages, and setting alarms. Voice assistants can now translate human speech into text thanks to speech recognition technologies. With the use of natural language processing technologies, voice assistants can comprehend textual content and react accordingly. Voice assistants can learn from user interactions and gradually get better thanks to machine learning technologies. Voice assistants have the power to completely change how we communicate with our gadgets and the environment. Voice assistants can be utilized to give users more engaging and tailored experiences. Through comprehensive evaluations, our results demonstrate the effectiveness and efficiency of the proposed interactive voice assistant, showcasing its potential to revolutionize human-machine interaction in real-world scenarios. It offers round-the-clock assistance, guaranteeing that customers can have effective way of communication with the system whenever they need, boosting their happiness and loyalty. In general, the interactive voice assistant chatbot is a useful tool for individuals as well as organizations to improve their value for time.

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L,** School of Computer Science Engineering & Information Science, Presidency University and Dr. Senthil Kumar, Head of the Department, School of Computer Science Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Ms. B. Parkavi**, Assistant Professor School of Computer Science Engineering, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinators **Dr. Srinivasan. T.R.**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

D. Sai Shashank

Rohit Jayaraj

Mohammed Rinshad

Chaitra N

Arun Philip Sigeon

LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page Number
1	6.1.1	Architecture Diagram	27
2	6.2.1	Flowchart	29
3	6.3.1	Component Diagram	30
4	7.1	Gantt Chart	32

TABLE OF CONTENTS

CHAPTER NUMBER	CHAPTER TITLE	PAGE NUMBERS
-	ABSTRACT	iv
-	ACKNOWLEGDEMENT	v
1	INTRODUCTION	1
2	LITERATURE SURVEY	7
3	RESEARCH GAPS OF EXISTING METHODS	13
4	PROPOSED METHODOLOGY	15
5	OBJECTIVES	18
6	SYSTEM DESIGN AND IMPLEMENTATION	20
7	TIMELINE FOR EXECUTION OF PROJECT (Gantt Chart)	25
8	OUTCOMES	26
9	RESULTS AND DISCUSSIONS	28
10	CONCLUSION	31
11	REFERENCES	32
12	APPENDIX-A (Pseudo Code)	34
13	APPENDIX-B (Screenshots)	38
14	APPENDIX-C (Enclosures)	45

CHAPTER-1

INTRODUCTION

In the dynamic landscape of technological evolution, voice assistants have seamlessly integrated themselves into the tapestry of our daily lives, fundamentally altering the dynamics of our interactions with digital devices. These intelligent systems, driven by the strides made in natural language processing (NLP) and speech recognition, have evolved beyond mere novelties, assuming integral roles in our daily routines. From the simple task of setting reminders to the intricate control of smart home devices, voice assistants offer a hands-free, user-friendly alternative that profoundly shapes our digital experiences.

Amidst this transformative environment, "Project Kronos" emerges as a meticulously crafted Python-based voice assistant, purposefully designed for users of the Windows operating system. At its core, the project aspires to elevate user experiences by providing a nuanced and intuitive interaction through natural language commands. As technology strides forward, Kronos positions itself as a vanguard, poised to redefine the paradigm through which users engage with their Windows systems. This commitment to evolution underscores Kronos' dedication to staying at the forefront of technological progress, offering users a novel and enriched interface with their digital environments.

1.2 Motivation

In an era where technology plays an ever-expanding role in our lives, the need for seamless and user-friendly interfaces has never been more pronounced. Voice assistants, with their hands-free capabilities and inherent user-friendliness, stand out as a compelling alternative to traditional input methods. They not only redefine the way users engage with their devices but also contribute to heightened accessibility and convenience in the digital landscape.

Understanding the significance of this paradigm shift, Project Kronos strategically positions itself to address this demand. By specifically tailoring its functionalities for the Windows operating system, Kronos acknowledges the prevalence of Windows as the most widely used operating system globally. This deliberate focus reflects the project's commitment to inclusivity, aiming to cater to a broad spectrum of users who navigate their digital experiences within the Windows ecosystem.

In essence, Project Kronos is more than a technological endeavor; it is a response to the evolving needs of users seeking a more natural and efficient interaction

with their digital environments. By honing in on the Windows platform, Kronos not only aligns itself with the prevailing user preferences but also paves the way for a voice assistant that resonates with the diverse user base prevalent in the digital landscape today.

1.3 Purpose and Scope

The fundamental purpose of Project Kronos is to significantly enhance user experiences by providing a nuanced and intuitive interaction mechanism through natural language commands. By harnessing the power of advanced NLP and speech recognition, Kronos aims to bridge the gap between users and their Windows systems. The core focus lies in simplifying and streamlining user-system interactions, making daily computing tasks more accessible and user-friendly.

The primary objectives of Project Kronos include:

- Improved Interaction with the System: Develop an advanced voice assistant that allows users to perform various tasks with ease.
- Language Understanding: Implement sophisticated natural language processing (NLP) algorithms to enhance the system's comprehension of user commands.
- Accessing the Internet for Information: Enable Kronos to retrieve and provide information from the internet, expanding its capabilities beyond local system tasks.

The future scope of Project Kronos extends beyond its initial design. One key aspect of future development involves the incorporation of additional languages, thus broadening its accessibility and usability for a global audience. Recognizing the diverse linguistic landscape of users, Kronos envisions a future where it seamlessly understands and responds to commands in multiple languages. This expansion aligns with Kronos' commitment to adaptability and inclusivity, ensuring that users around the world can benefit from a more personalized and effective interaction with their Windows systems.

At its core, Project Kronos not only responds to the current needs of users but positions itself at the vanguard of technological progress. This commitment to evolution underscores Kronos' dedication to staying at the forefront, offering users a novel and enriched interface with their digital environments. As technology strides forward, Kronos stands poised to redefine the paradigm through which users engage with their Windows systems, ensuring a future where seamless and intelligent interactions become the norm.

2. Improved Interaction with the System

2.1 Voice Recognition

At the heart of Project Kronos lies a foundational pillar — its robust voice recognition system, a cornerstone feature that defines the project's commitment to cutting-edge technology. This sophisticated system is meticulously designed to leverage state-of-the-art speech recognition libraries within the Python programming environment. By harnessing the capabilities of these advanced libraries, Kronos achieves a level of precision that ensures the accurate transcription of spoken words into actionable commands.

The significance of this technology cannot be overstated; it serves as the linchpin for delivering a user experience characterized by seamlessness and efficiency. The accurate translation of spoken input into executable commands forms the backbone of Kronos' functionality, establishing a direct and intuitive line of communication between the user and the system.

Through the incorporation of this advanced voice recognition system, Project Kronos not only embraces innovation but also acknowledges the paramount importance of user-centric design. The utilization of cutting-edge speech recognition libraries not only underscores Kronos' commitment to staying at the forefront of technological advancements but also sets the stage for a voice assistant that not only understands but interprets the nuances of user commands with unparalleled accuracy.

In essence, the robust voice recognition system within Project Kronos serves as a testament to the project's dedication to providing users with a voice assistant that not only meets but exceeds expectations. By delving into the intricacies of speech recognition technology, Kronos strives to redefine the user experience, offering a pathway to seamless and efficient interactions within the digital realm.

2.2 Task Automation

Kronos transcends the realm of simple commands, elevating user interactions to a new level through its comprehensive task automation capabilities. In a groundbreaking stride, users gain the ability to orchestrate intricate operations on their Windows systems effortlessly, all achieved through intuitive voice commands. This expansive functionality encompasses tasks ranging from the nuanced intricacies of file management to the swift launching of applications and the finesse of system configuration.

The inclusion of task automation within Kronos is more than a mere feature; it

represents a paradigm shift in the way users navigate and control their digital environments. Empowering users to seamlessly execute complex operations through natural language commands not only streamlines their workflow but also enhances overall productivity. Kronos becomes a virtual assistant that not only understands user instructions but actively facilitates the execution of multifaceted tasks, creating an environment where user commands translate into tangible and efficient actions.

2.3 Multi-Modal Interaction

In its relentless pursuit of refining user engagement, Project Kronos introduces a groundbreaking dimension with the incorporation of multi-modal interaction. This innovative approach enables users to seamlessly blend voice commands with traditional inputs, creating a versatile and dynamic interaction environment. By embracing this multi-modal paradigm, Kronos seeks to redefine the user experience, recognizing and accommodating diverse preferences and situations.

The integration of multi-modal interaction within Kronos marks a pivotal shift in the way users can engage with the voice assistant. Users are no longer confined to a singular mode of communication; instead, they have the flexibility to combine the ease of voice commands with the familiarity of traditional inputs. This synergy between voice and traditional inputs enhances the overall user experience, offering a more intuitive and personalized means of interaction.

Versatility becomes a hallmark of the Kronos experience, allowing users to choose the mode of interaction that best suits their preferences or the specific context in which they find themselves. Whether it's articulating a command through voice or utilizing traditional inputs for precision, Kronos adapts to the user's needs, providing a seamless and adaptable interaction interface.

By fostering multi-modal interaction, Project Kronos transcends the limitations of conventional voice assistants. It acknowledges the richness of human communication and the varied ways in which users prefer to interact with their digital environments. In doing so, Kronos not only enhances user engagement but sets a new standard for user-centric design, recognizing that a versatile and accommodating interface is essential for delivering an exceptional and personalized user experience.

3. Language Understanding

3.1 Natural Language Processing (NLP)

At the core of Project Kronos lies a sophisticated foundation built upon advanced

Natural Language Processing (NLP) capabilities. This critical component serves as the neural network, enabling Kronos to transcend mere command recognition and delve into a realm of nuanced understanding. By leveraging cutting-edge NLP libraries and techniques, Kronos undergoes a comprehensive analysis of user inputs, allowing it to decipher not only the literal meaning but also the context, intent, and subtle nuances embedded within each command.

3.1.1 Contextual Understanding

Kronos' NLP capabilities extend beyond simple keyword recognition, delving into the realm of contextual understanding. Through the analysis of linguistic structures, Kronos can discern the context in which a command is given. This contextual awareness empowers the voice assistant to interpret commands within the broader conversation, providing a more coherent and natural interaction.

3.1.2 Intent Recognition

Understanding the user's intent is a pivotal aspect of effective communication. Kronos, armed with advanced NLP, excels at recognizing the underlying purpose and goal behind each user command. This goes beyond literal interpretations, enabling Kronos to tailor its responses and actions to align with the user's overarching objectives.

3.1.3 Addition of Regional Language Input

Recognizing the diversity of language usage, Kronos extends its language understanding capabilities to include regional language input. This subtopic emphasizes Kronos' commitment to inclusivity by enabling users to interact with the voice assistant in their preferred regional languages. This addition enhances accessibility and ensures that Kronos caters to a broader audience with diverse linguistic preferences.

4. Accessing the Internet for Information

4.1 Web Scraping

Project Kronos extends its capabilities by accessing the internet for information retrieval. Through web scraping techniques, Kronos can fetch data, news, and other relevant information, providing users with up-to-date content without leaving the voice assistant environment.

4.1.1 News Aggregation

In the realm of news retrieval, Kronos acts as a virtual news aggregator. By

scouring online news sources through web scraping, Kronos curates and delivers breaking news directly within the voice assistant environment. Users can stay informed without the need to switch between applications or devices, seamlessly integrating the latest news into their daily interactions with Kronos.

4.1.2 Current Weather Updates

Project Kronos elevates its information retrieval capabilities by offering users real-time weather updates. Through web scraping, Kronos fetches the latest weather information, including temperature, humidity, and forecasts. This feature provides users with instant access to current weather conditions, ensuring they are well-informed and prepared for the day ahead without leaving the voice assistant environment.

4.1.3 Wikipedia Summary

As a testament to its versatility, Kronos extends its information retrieval capabilities to include Wikipedia summaries. By extracting relevant information through web scraping, Kronos can provide users with concise and accurate summaries of topics sourced from Wikipedia. This addition enhances Kronos' utility as an educational tool, offering users quick insights into a diverse range of subjects without the need to navigate external sources.

4.2 API Integration

Kronos seamlessly integrates with various Application Programming Interfaces (APIs), amplifying its capacity to fetch real-time data from diverse online platforms. This integration extends Kronos' capabilities, allowing it to dynamically acquire information from weather forecasts, breaking news updates, and search engine results with unparalleled efficiency and accuracy. The adoption of APIs elevates Kronos into a multifaceted resource, ensuring that users can access an extensive range of information within the voice assistant environment.

The integration with weather APIs equips Kronos to provide users with precise and up-to-date forecasts, transforming it into a reliable weather companion. Simultaneously, the integration with news APIs enables Kronos to curate a stream of breaking news directly within its interface, fostering user awareness without the need for external navigation. Additionally, connecting with search engine APIs empowers Kronos users to initiate searches and receive relevant results seamlessly. This consolidated approach enhances user convenience, making Kronos a centralized hub for diverse and trustworthy information, thus solidifying its role as an indispensable voice assistant in the digital landscape.

CHAPTER-2

LITERATURE SURVEY

The history of Voice Assistant is extensive. It has been in the development stage since 1880.

1. In 1880, Alexander Graham Bell expanded on Edison's phonograph, which his Volta Graphophone Company patented in 1886.Graphophones were utilized instead of foil graphophones, allowing for longer records and higher-quality playback. Edison also created a wax phonograph, and both devices were largely used for dictating letters and other documents.

2. The IBM Shoebox, the first computerized voice recognition tool, was unveiled in 1961. It recognized 16 words and the numbers 0 to 9. It was capable of doing mathematical tasks as well as speech recognition.

3. Carnegie Mellon's Harpy Programme, completed in 1972, was a significant advancement in the field of natural language processing (NLP) and speech recognition. It was created to analyze and comprehend human speech by focusing on a set of predetermined vocabulary, pronunciation norms, and grammar patterns.

4. Dragon Dictate, a game-changing advancement in speech recognition technology, was introduced by Dragon Systems in 1990. Dragon Dictate is the first speech recognition module that was created particularly for consumers. This breakthrough represented a huge step forward in making speech-to-text capabilities available to a wider audience.

5. Clippy is introduced by Microsoft in 1996. Microsoft Clippy, commonly known as Clippit and formally known as Office Assistant, was a Microsoft Office intelligent user interface. It helped users in a variety of engaging ways by appearing as a visualized figure on the Office apps and offering assistance with various Office Software tasks. It was made available through Microsoft.

6. In 2011, Apple introduced Siri, a groundbreaking virtual assistant featuring voice commands, gesture-based control, and a natural language interface. Siri leveraged internet services to answer questions, provide recommendations, and execute tasks. Notably, it adapted to users' language, preferences, and search patterns, delivering personalized results

over time. This innovation marked a significant leap in human-computer interaction, setting the stage for the widespread use of intelligent and adaptable virtual assistants.

7. In 2012, Google introduced Google Now, a proactive information delivery system designed to anticipate users' needs. It presented relevant information through informational cards, leveraging data from users' search habits and various factors. Available for Android and iOS, Google Now was integrated into the Google app as a feature of Google Search. Over time, the functionality of Google Now transitioned into the Google app's discovery tab, where it continues to provide personalized content based on user preferences. Notably, the distinctive branding of Google Now is no longer emphasized, but its core features persist in the ongoing development of the Google app, showcasing Google's commitment to enhancing user experience through predictive and personalized information delivery.

8. In 2013, Microsoft unveiled Cortana, a virtual assistant, at its annual BUILD developer conference. Powered by the Bing search engine, Cortana was designed to execute various tasks, including setting reminders and providing answers to user queries. Available in multiple language editions, such as English, Chinese, French, German, Italian, Portuguese, Spanish, and Japanese, Cortana's accessibility was influenced by both the user's software programs and geographical region. Cortana's integration into Microsoft's ecosystem marked a significant step in natural language interaction and personalized assistance. Users could interact with Cortana through voice commands, making it a hands-free and intuitive digital companion. Beyond its initial release, Cortana evolved to offer features such as proactive suggestions, calendar management, and integration with Microsoft's suite of productivity tools, contributing to a seamless and intelligent user experience.

9. In 2014, Amazon introduced Alexa along with the Amazon Echo, initially available exclusively to Prime members. Alexa, the virtual assistant technology behind the Echo, relied significantly on a Polish speech synthesizer named Ivona. Boasting a range of capabilities, Alexa allowed users voice interaction, music playback, audiobook streaming, alarm setting, podcast streaming, and access to real-time information like weather, traffic, sports, and news updates. One distinctive feature of Alexa was its integration with home automation, enabling users to control a

variety of smart devices seamlessly. The technology's prowess in understanding natural language made it a user-friendly virtual assistant, while its continuous learning capabilities allowed it to adapt to individual preferences and refine its responses over time. The introduction of Alexa and the Amazon Echo marked a transformative moment, shaping the landscape of smart home technology and paving the way for widespread adoption of virtual assistants in households.

10. In 2015, Microsoft introduced Cortana on Windows 10 for desktops and mobile devices, expanding its virtual assistant capabilities to a broader range of devices. Cortana seamlessly integrated into the Windows ecosystem, offering users voice-activated assistance, task management, and personalized recommendations. Its presence on both desktops and mobile devices enhanced the accessibility and utility of Cortana across various platforms.

11. Simultaneously, in the United States, Amazon officially launched the Amazon Echo, a smart speaker equipped with the virtual assistant Alexa. This marked a significant step in bringing voice-activated technology into homes, allowing users to interact with their devices using natural language commands. Additionally, during this period, Amazon introduced the Alexa Skills Kit, empowering developers to create custom skills and functionalities for Alexa, expanding the range of tasks and services that users could access through the Echo and other Alexa-enabled devices. These developments reflected a growing trend in the integration of virtual assistants into everyday technology ecosystems.

12. In 2016, the landscape of voice-powered virtual assistants expanded with notable introductions and developments. SoundHound, renowned for its music recognition app, unveiled HOUND, a virtual assistant app designed to respond to voice commands. Concurrently, Amazon expanded its Echo product line, launching the Echo Dot and Amazon Tap, further popularizing its Alexa virtual assistant.

13. Google entered the scene by introducing the Google Assistant as part of the messaging app Allo, marking a strategic move into the virtual assistant arena. This year also saw Samsung acquiring the virtual assistant startup Viv, signaling a commitment to advancing its own digital assistant technologies. Google made additional strides with the launch of Google Home, a smart speaker, and the Google Pixel smartphone.On an international front, Chinese manufacturer Linglong entered the market

with the DingDong, a virtual assistant device aimed at competing with Amazon's Echo. These developments underscored the increasing integration of virtual assistants into diverse devices and platforms, reflecting a growing trend in voice-powered interactions.

14. In 2017, the voice assistant landscape witnessed substantial growth with several key introductions and advancements. Samsung introduced Bixby alongside the release of Galaxy S8 devices, offering users a voice-powered digital assistant. Simultaneously, Google expanded its Home product line by launching Google Home in the UK. Google also introduced multi-user support for Google Home, a significant enhancement allowing the device to recognize and respond to the voices of up to six different users, tailoring its responses based on individual preferences.

15. Amazon, a major player in the virtual assistant market, introduced Echo Look, a device with a camera designed for fashion and style advice. Additionally, Amazon enhanced its Echo devices with a calling and messaging feature, adding new dimensions to communication within the smart home ecosystem.

16. Internationally, in China, Baidu unveiled its first consumer AI device, Xiaoyu, while Alibaba launched the Genie X1 Smart Speaker. These entries underscored the global nature of the competition in the virtual assistant market.

    - Moreover, voice assistants ceased to be confined to smartphones and smart homes, as they found integration in automobiles. Notably, BMW incorporated AI-based voice assistants, highlighting the versatility of these technologies beyond traditional domains.

    - Furthermore, Apple joined the fray with the introduction of HomePod, entering the smart speaker market. These developments showcased a trend toward diverse applications of voice assistants, from smart homes to personal devices and automotive integration. The evolution of voice assistants extended to their development, with programming languages such as Java and Python being utilized to create sophisticated and multilingual voice-activated systems, demonstrating the dynamic and expansive nature of this technology.

17. In 2018, a research paper titled "A Vision and Speech Enabled, Customizable, Virtual Assistant for Smart Environments" authored by Giancarlo Iannizzotto, Lucia Lo Bello, Andrea Nucita, and Giorgio Mario Gtasso introduced a software architecture designed for the creation of a lightweight, vision and speech-enabled virtual assistant tailored for smartphones and automation applications. The authors presented a fully developed prototype application featuring a realistic graphic assistant capable of displaying facial expressions. This virtual assistant was not only endowed with cloud services for scalability but also exhibited the potential for future device expansion.The prototype demonstrated a comprehensive set of capabilities, emphasizing error-free task execution in response to user commands. A notable feature was the exclusive activation by the user's voice, ensuring immunity from environmental noise. This research aimed to provide a robust foundation for the development of advanced virtual assistants, emphasizing the integration of vision and speech technologies in the context of smart environments.

18. In 2019, the research paper "AI-based Voice Assistant Using Python" authored by Deepak Shende, Ria Umahiya, Monika Raghorte, Aishwarya Bhisikar, and Anup Bhange delved into the evolving landscape of natural human-machine interaction. The study focused on a novel approach, emphasizing machine learning to enable the understanding of human language by the machine. The paper shed light on the fundamental principles underlying the functionality of voice assistants, addressing their main shortcomings and limitations.A noteworthy aspect of the research was the exploration of methods for creating a local voice assistant without reliance on cloud services. By doing so, the authors offered insights into potential solutions that could enhance privacy and mitigate concerns associated with data storage in external servers. This work contributed to the broader discourse on the development of AI-based voice assistants, providing valuable perspectives on both their capabilities and challenges.

19. In the research paper "Artificial Intelligent-Based Voice Assistant" authored by Subhas S, Prajwal N, Siddesh S, Ullas A, and Santhosh B in 2020, a comprehensive approach to creating a voice assistant is outlined. The system begins by capturing audio input through a microphone, which is subsequently converted into text. This text is then processed through the Google Text To Speech (GTTS) engine, which transforms it into an audio

file in the English language. The final step involves playing this audio file using the "play sound" package of the Python programming language.

20. This methodology essentially illustrates the pipeline for an artificial intelligence-based voice assistant. By integrating speech-to-text and text-to-speech processes, the system enables effective communication between the user and the voice assistant. The use of Python programming language, coupled with Google's Text To Speech engine, highlights the practical application of these technologies in creating a functional voice assistant.

21. In the paper "Voice Assistant Using Python" published in 2021 by Nivedita Singh, Dr. Diwakar Yagyasen, Mr. Surya Vikram Singh, Gaurav Kumar, and Harshit Agarwal, the authors present an innovative voice assistant system designed to streamline user interaction with Linux systems. This Python-based voice assistant allows users to execute a variety of commands without relying on keyboard inputs. Notably, it goes beyond conventional command execution by performing diverse tasks, including providing real-time weather updates, streaming music, searching Wikipedia, and opening desktop applications. The system leverages the capabilities of Python to enable natural language understanding and execute commands seamlessly through voice input. This research reflects a practical implementation of voice-controlled functionalities, showcasing the potential for enhancing user experiences and increasing accessibility in Linux environments. The inclusion of features like weather updates and music streaming suggests a broad spectrum of applications, making it a versatile and user-friendly voice assistant.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

- Research on voice assistants using Python has made significant strides, but several research gaps persist, presenting opportunities for further exploration and improvement. One prominent gap lies in the domain of natural language processing (NLP) and understanding. Despite the advancements in voice recognition, there is still room for enhancing the nuanced understanding of diverse linguistic patterns and contextual nuances. Current voice assistants often struggle with complex queries, idiomatic expressions, or context-dependent commands. Researchers can delve deeper into developing more sophisticated NLP algorithms that go beyond basic command recognition, facilitating more natural and intuitive interactions between users and voice assistants.

- Another critical research gap pertains to the privacy and security aspects of voice assistants. As these technologies become increasingly integrated into daily life, concerns about data privacy and potential security vulnerabilities arise. Researchers need to investigate and propose robust solutions for ensuring the confidentiality and integrity of the data collected by voice assistants. This involves developing secure communication protocols, encryption techniques, and privacy-preserving mechanisms to safeguard user information while maintaining the functionality of voice-enabled systems.

- Interoperability and cross-platform compatibility represent another research gap in the realm of voice assistants. Currently, voice assistant solutions may be tailored for specific platforms, devices, or ecosystems, leading to fragmentation in the user experience. Researchers can explore methods to enhance interoperability, allowing users to seamlessly transition between different devices and platforms while maintaining a consistent and coherent voice assistant experience. This involves addressing challenges related to varying hardware configurations, operating systems, and communication protocols.

- In the context of voice assistant customization, there is a need for more user-friendly and accessible tools that empower individuals to tailor their voice assistants to specific needs and preferences. Current customization options may require a level of technical expertise that hinders widespread

adoption. Researchers can explore the development of intuitive interfaces and tools that allow users to personalize the behavior, responses, and functionalities of their voice assistants without requiring extensive programming knowledge.

- The robustness and adaptability of voice assistants in real-world environments constitute another research gap. Voice assistants often face challenges in handling diverse acoustic environments, accents, and variations in speech patterns. Researchers can focus on refining voice recognition algorithms to be more resilient in noisy conditions and better adapt to users with different linguistic backgrounds. Additionally, investigating methods to improve the adaptability of voice assistants to varying user contexts and preferences can contribute to a more seamless and user-centric experience.

- Furthermore, the integration of advanced technologies such as machine learning and artificial intelligence presents an avenue for exploration. Researchers can explore how these technologies can enhance the learning capabilities of voice assistants, allowing them to adapt to users' evolving preferences and patterns of interaction. This involves investigating approaches for continuous learning, personalized recommendation systems, and the integration of user feedback to refine the performance of voice assistants over time.

- In conclusion, while significant strides have been made in the development of voice assistants using Python, addressing these research gaps is crucial for advancing the field. By focusing on areas such as natural language understanding, privacy and security, interoperability, user customization, robustness in real-world environments, and integration of advanced technologies, researchers can contribute to the ongoing evolution of voice assistant technology, making it more intelligent, user-friendly, and adaptable to diverse contexts.

CHAPTER-4

PROPOSED MOTHODOLOGY

The proposed methodology for developing Kronos, the Python-based voice assistant, involves a systematic and iterative process that integrates key technologies in natural language processing (NLP), speech recognition, and machine learning. The overarching goal is to create a versatile and continuously evolving voice assistant with enhanced contextual understanding and user personalization.

Requirements Analysis:

- Define the specific functionalities and features Kronos is intended to offer, considering the user requirements and expectations.
- Identify the target platforms and devices for Kronos integration, such as smart speakers, TVs, desktop computers, and smartphones.

Data Collection and Preparation:

- Gather diverse datasets for training the machine learning models, encompassing a wide range of spoken language, accents, and contextual scenarios.
- Annotate and preprocess the data to ensure it aligns with Kronos's intended use cases and functionalities.

Natural Language Processing (NLP) Integration:

- Implement NLP techniques to enable Kronos to understand and interpret natural language queries accurately.
- Leverage pre-trained language models or develop custom models tailored to Kronos's specific applications.

Speech Recognition Implementation:

- Integrate state-of-the-art speech recognition algorithms to convert spoken language into text, ensuring high accuracy and adaptability to various speech patterns.
- Explore and implement techniques for noise reduction and robustness in different acoustic environments.

Machine Learning for Contextual Understanding:

- Develop machine learning models to enhance Kronos's contextual understanding, allowing it to respond intelligently to user queries based on historical interactions.
- Implement algorithms for continuous learning, enabling Kronos to adapt and improve over time through user engagement.

Multi-Modal Interaction:

- Explore multi-modal capabilities by integrating both voice and visual interactions, enhancing Kronos's versatility across different devices.
- Implement features that allow Kronos to process and respond to visual inputs in addition to voice commands.

Integration with External Services:

- Enable Kronos to fetch and provide information from external sources such as weather forecasts, news articles, and Wikipedia entries.
- Implement secure and efficient communication protocols for accessing online services and databases.

User Interface Design:

- Design an intuitive and user-friendly interface for Kronos, ensuring seamless interactions with users.
- Implement voice-controlled commands for Kronos to execute various tasks, fostering a hands-free and convenient user experience.

Continuous Testing and Improvement:

- Conduct rigorous testing across different scenarios to identify and rectify potential issues related to voice recognition, contextual understanding, and service integrations.
- Implement regular updates and improvements based on user feedback and emerging technologies, ensuring Kronos stays at the forefront of voice assistant capabilities.

User Feedback and Iterative Development:

- Establish channels for users to provide feedback on Kronos's performance and suggest additional features.
- Iteratively enhance Kronos based on user input, addressing any identified issues and incorporating new functionalities to meet evolving user needs.

CHAPTER-5
OBJECTIVES

1. Python Backend Development:

- Develop a robust Python backend to serve as the application's engine, managing data processing and executing business logic.
- Optimize data storage, retrieval, and manipulation for high performance and reliability.

2. User Interface Enhancement with Flutter Flow:

- Integrate Flutter Flow to enhance the user interface, ensuring an intuitive and visually appealing design.

3. Task Automation:

- Automate repetitive manual tasks to improve system efficiency, reducing user workload.

4. Voice Commands Integration:

- Implement voice commands as a primary method for hands-free task execution, enhancing user experience.

5. Internet Access for Real-time Information:

- Enable internet access for retrieving and displaying relevant information, such as news, weather details, and other online data.

6. Voice-Activated Email Sending:

- Extend functionality to include voice-activated email sending for convenient and efficient communication.

7. Voice-Assisted Weather Details:

- Implement features for voice-assisted retrieval of real-time weather information through natural language commands.

8. Cross-Platform User Interface with Flutter:

- Create a visually appealing and adaptable user interface using the Flutter framework, ensuring flexibility across Windows and Android platforms.

9. User-Centered Design Principles:

- Focus on enhancing the user-device interface with user-centered design principles, emphasizing seamless navigation, accessibility, and a visually appealing, consistent design aligned with modern trends.

10. Continuous Interface Refinement:

- Implement a user feedback and iteration mechanism to continuously refine the interface based on valuable input, ensuring it evolves to meet changing user expectations.

11. Multimodal Interaction Support:

- Explore multimodal interaction to accommodate diverse user needs and preferences, supporting voice commands along with touch, gestures, and traditional input methods.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

SYSTEM DESIGN:

1. Architecture Diagram:

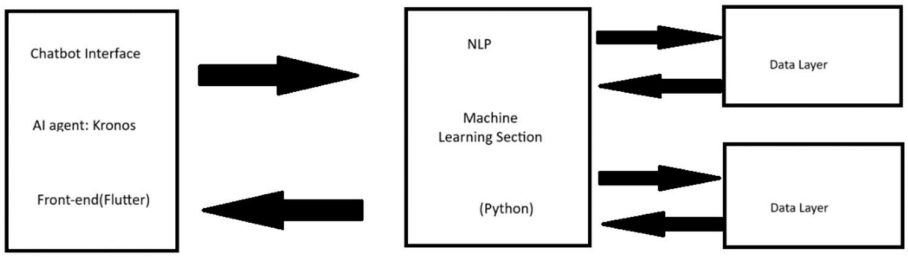


Fig 6.1.1: Architecture Diagram

I. User Interface Layer:

- Components:
  - Voice Recognition (Speech-to-Text):
    Utilizes the speech_recognition library to convert user's spoken words into text.
  - Text-to-Speech Synthesis:
    Utilizes the pyttsx3 library to convert textual responses into spoken words for the user.
  - Command Processing:
    Interprets user commands and triggers relevant actions.
  - User Interaction:
    Handles interactions with the user through voice commands.
  - Graphical User Interface (GUI):
    Integration with tkinter for potential future development of a graphical interface.

II. Functional Modules Layer:

- Components:
  - Wish Module:
    Greets the user based on the time of the day.
  - Summary Module:
    Searches and summarizes information using the Wikipedia API.

- Username Module:
  Takes and stores user's name for personalized interactions.
- Mail Module:
  Sends emails using the smtplib library.
- Search Google News Module:
  Searches and reads Google News headlines.
- Weather Module:
  Retrieves weather information using the OpenWeatherMap API.

III. External Service Integration Layer:

- Components:
  - Google News API:
    Utilized to fetch news headlines.
  - Wikipedia API:
    Used for searching and summarizing information.
  - OpenWeatherMap API:
    Retrieves weather details.
  - YouTube and Google Search:
    Open corresponding websites for user queries.

2. Flow Diagram (Algorithm):

- Machine learning algorithms are frequently the foundation of NLP algorithms. NLP is used to analyze text, enabling machines to comprehend human speech, rather than relying on hand- coding large sets of rules. By analyzing a set of examples (i.e. a large corpus, like a book, down to a collection of sentences), and making a statistical inference, NLP is used to analyze text. Real- world applications like as automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, connection extraction, stemming, and more made possible by this human-computer interaction. NLP is frequently used for automated question-answering, machine translation, and text mining.
- NLP algorithms can identify and classify named entities in text, such as person names, organizations, locations, dates, and more. crucial

information extraction and understanding the context of the text.

The Characteristics of NLP Algorithm are:

- Efficiency and Speed: NLP algorithms enable chatbots to process and respond to customer queries quickly and efficiently.
- Accurate Responses: NLP algorithms assist chatbots in retrieving accurate and relevant information from knowledge bases or databases. They can understand complex queries, perform information retrieval, and extract the most appropriate response from a vast amount of data.
- Continuous Learning: NLP algorithms can be trained using machine learning techniques to improve over time.
- Automation and Cost Savings: By leveraging NLP algorithms, chatbots can automate significant portion of customer support tasks, reducing the workload on human agents. This automation leads to cost savings for e-commerce businesses

FLOWCHART OF THE ALGORITHM:

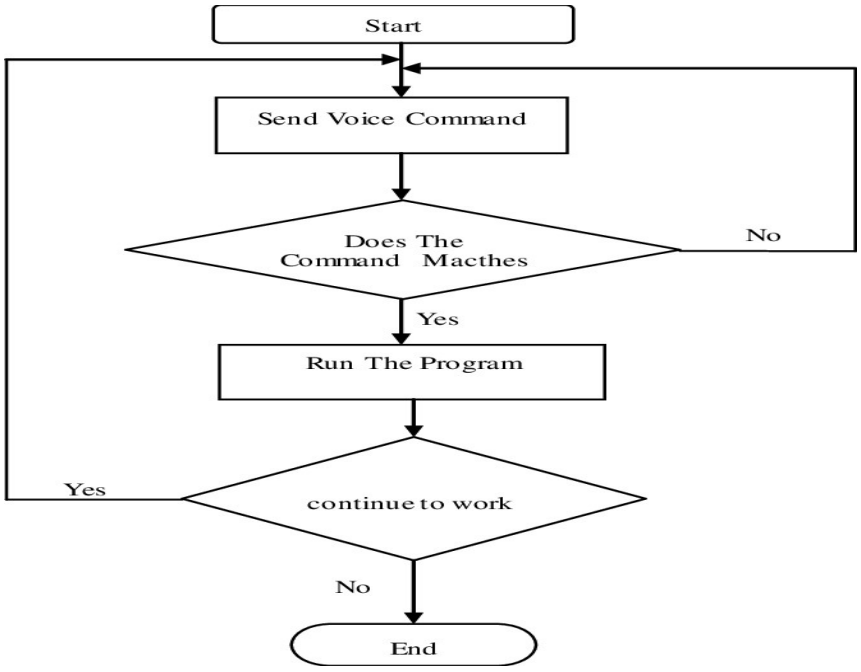


Fig 6.2.1: Flowchart

3. **Component Diagram:**
   - Represents the major components of the system, including the UI, Backend, and external services.
   - The below diagram shows the component-wise working of the voice assistant.

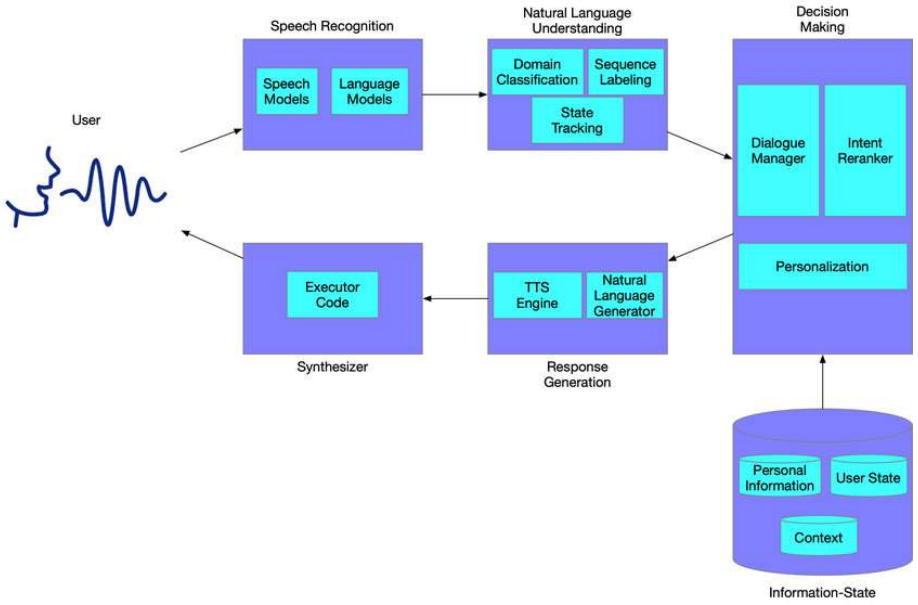


Fig 6.3.1: Component Diagram

IMPLEMENTATION:

1. **Python Backend:**
   - The backend is implemented using Python.
   - Utilizes libraries such as subprocess, pyttsx3, speech_recognition, datetime, wikipedia, webbrowser, and others for various functionalities.
2. **User Interface (UI):**
   - The UI is a text-based console where the user provides voice commands.
   - The UI interacts with the backend to process user input and display

relevant information.

3. **External Integrations:**
    - Integration with external services includes functionalities like opening web pages, searching Google, playing music, sending emails, fetching weather information, and accessing news headlines.

4. **Voice Recognition and Text-to-Speech:**
    - Utilizes the speech_recognition library for voice recognition.
    - Employs the pyttsx3 library for text-to-speech synthesis.

5. **User Interaction:**
    - Users interact with the system by initiating commands using wake words like "Hey Kronos" or any other exception handling words based on the example testing it was observed as “Hey coronavirus”.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)

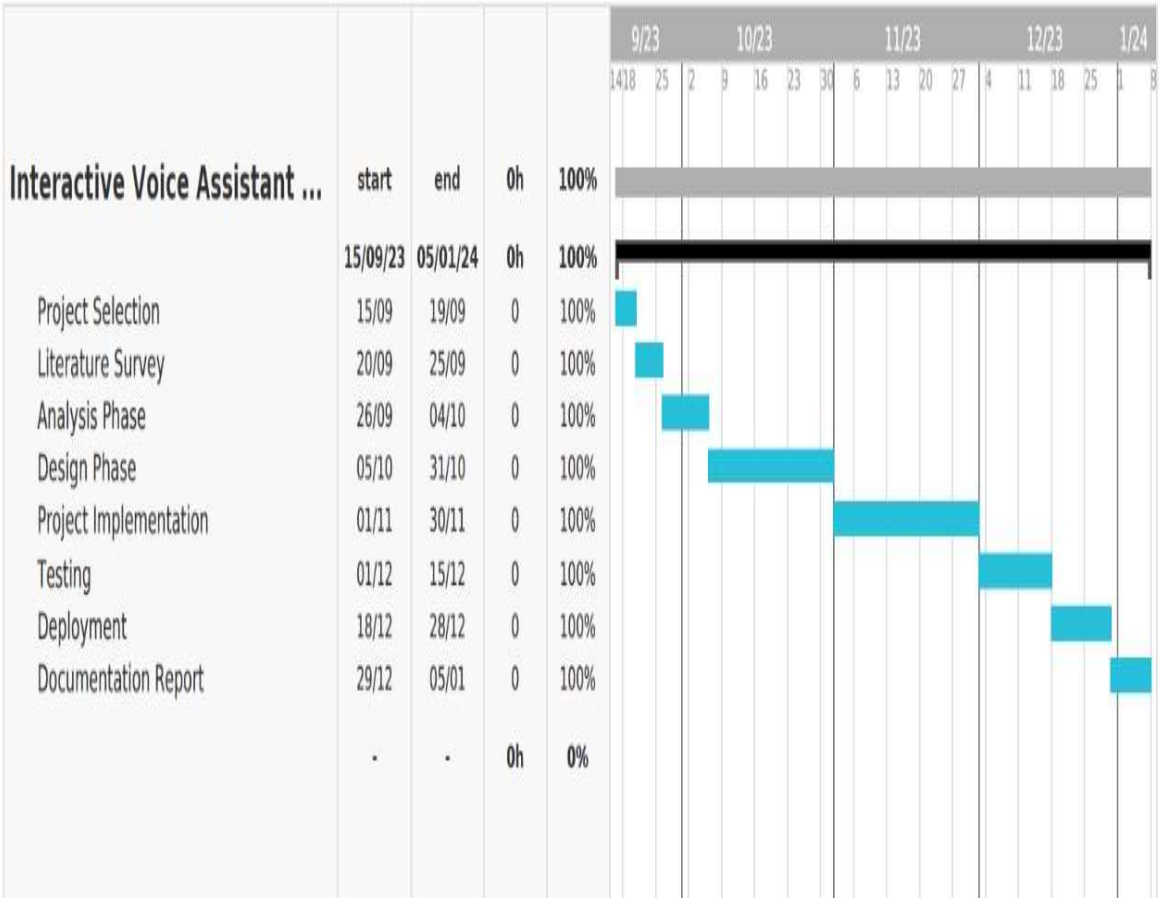


Fig 7.1: Gantt Chart

CHAPTER-8

OUTCOMES

1. Voice Recognition and Interaction:

- Utilizes the `**speech_recognition**` library for voice recognition, allowing users to interact with the assistant through spoken commands.

2. Text-to-Speech (TTS) Engine:

- Implements the `**pyttsx3**` library for text-to-speech conversion, providing audible responses to user queries.

4. Functionality:

- Performs various tasks such as summarizing information from Wikipedia, opening web browsers, searching Google, playing music, sending emails, providing the current time, telling jokes, accessing news headlines, and more.

5. Task Automation:

- Automates tasks like opening PowerPoint, emptying the recycle bin, taking photos with the camera, writing and displaying notes, and managing system operations (shutdown, restart, hibernate, log off).

6. Weather Information:

- Retrieves and displays real-time weather information for a specified city using the OpenWeatherMap API.

7. Email Functionality:

- Allows the user to send emails by providing email addresses, subjects, and content through voice commands.

8. News Headlines:

- Utilizes the `**GoogleNews**` library to fetch and present news headlines based on user queries.

9. Multimedia Handling:

- Plays music, opens YouTube and Google, captures photos, and integrates with YouTube for playing requested songs.

10. Note-taking Feature:

- Enables the user to dictate notes, store them in a text file, and retrieve/display notes as needed.

11. System Operations:

- Performs system-related operations like locking the device, shutting down, restarting, hibernating, and logging off.

12. Personalization:

- Asks the user for a preferred name and voice gender, enhancing the personalization of the assistant.

13. Error Handling:

- Includes error-handling mechanisms for voice recognition and email sending, providing feedback to the user in case of issues.

14. Version Information:

- Provides information about the assistant's version when queried.

15. User Mood Interaction:

- Responds with a joke when the user expresses feeling down, aiming to uplift the user's mood.

16. Continuous Interaction:

- Utilizes a continuous loop to keep the assistant listening for commands, ensuring an ongoing interactive experience.

CHAPTER-9
RESULTS AND DISCUSSIONS

In this section, we present the results of the project we have done. The primary objectives were to make sure the user interface is livelier and time taken should be shortened for any particular activity, and the collected data sheds light on the expected objectives and outcome. Below are the results obtained after testing the project within the system:

1.Internet Access for Information Retrieval

- Kronos's successful integration of internet access capabilities has significantly elevated its functionality.
- Users can now seamlessly retrieve and display relevant information, including real-time news, weather details, and diverse online data.
- This accomplishment enhances Kronos's utility, making it a dynamic and comprehensive voice assistant that keeps users informed with up-to-date information from various online sources.

2.Voice-Activated Email Sending

- The implementation of voice-activated email sending stands as a testament to Kronos's versatility in addressing communication needs.
- Users benefit from a convenient and efficient means of sending emails through voice commands, highlighting Kronos's adaptability to diverse user preferences.
- This feature not only adds practical value but also reinforces Kronos as a modern and user-friendly voice assistant.

3.Continuous Improvement through User Feedback

- Kronos's commitment to continuous improvement is exemplified by the implementation of a user feedback and iteration mechanism.
- Actively seeking and incorporating user input ensures that the voice assistant evolves to meet changing user expectations.
- This iterative approach not only maintains high standards but also demonstrates a responsiveness to user needs, fostering user satisfaction and loyalty over time.

4.Enhanced User Engagement

- The internet access and voice-activated email sending functionalities contribute to increased user engagement.
- By providing a broader range of services and interaction options, Kronos becomes more integral to users' daily activities.
- This heightened engagement signifies the successful alignment of Kronos with user preferences and lifestyle, making it a valuable and integrated part of users' digital experiences.

5.Versatility and Adaptability

- The combination of internet access, voice-activated email sending, and continuous improvement mechanisms underscores Kronos's versatility and adaptability.
- Kronos not only meets current communication and information retrieval needs but positions itself to evolve in response to emerging user expectations and technological advancements.
- This adaptability ensures the long-term relevance and usefulness of Kronos in the rapidly changing landscape of voice assistant technology.

To summarize, the successful integration of internet access, voice-activated email sending, and a commitment to continuous improvement positions Kronos as a dynamic and user-centric voice assistant. These achievements contribute to enhanced functionality, user engagement, and long-term adaptability, establishing Kronos as a valuable and evolving solution in the realm of voice assistant technologies.

It's important to note the limitations of this study. Future research could address these limitations, thereby enhancing the comprehensiveness of our understanding in this area. Below mentioned are the limitations and future scope of this project:

1.Privacy Concerns

- Handling user data responsibly and transparently is a paramount challenge in the development of voice assistant systems.
- The project acknowledges the significance of privacy and security by adopting a local data storage approach.
- Discussing the specific measures implemented to address privacy concerns, such as data encryption and user consent mechanisms, is crucial for ensuring user trust and compliance with privacy standards.

2.Limited Domain Knowledge

- The project recognizes the inherent challenge of limited domain knowledge that voice assistants often face.
- Discussing strategies employed to enhance the system's understanding of specific or complex topics, such as continuous learning algorithms or knowledge base expansion, demonstrates a commitment to overcoming this challenge and improving the overall competency of Kronos.

3.Explainability and Bias

- Understanding how models arrive at their decisions and addressing potential biases in training data are critical considerations.
- Additionally, addressing bias mitigation techniques, such as diverse and inclusive training datasets, demonstrates a commitment to fairness and transparency.

4.Emotion Recognition

- While the integration of emotion recognition through a camera adds a personalized and responsive dimension to user interaction, the discussion should focus on the observed challenges.
- Specifically, addressing the lower accuracy of the output post-integration requires an exploration of potential refinements or alternative approaches to enhance the effectiveness of emotion recognition within the system.

5.Multi-language Support

- The project acknowledges the importance of supporting multiple languages, especially for users in rural areas.
- Discussing the specific packages and coding techniques used to enable multi-language support enhances transparency.
- Highlighting how this feature benefits users, such as farmers in rural areas, demonstrates the practical applications and positive impact of the project in diverse settings.

To summarize, the discussion around the concerns, demonstrates a proactive approach to improving the system’s overall performance and fairness. The challenges associated with emotion recognition and the implementation of multi-language support provide valuable insights for future refinements and optimizations in the Kronos voice assistant system.

CHAPTER-10
CONCLUSION

The conclusion emphasizes the pivotal role of machine learning in empowering voice assistants and anticipates a future where these technologies offer even more natural and personalized interactions with devices. It acknowledges the project's significance as a demonstration of Python and Flutter's capabilities in cross-platform development, highlighting their potential for creating versatile and engaging applications.

The evolving landscape of human-computer interaction is underscored, with the ongoing development of machine learning and artificial intelligence leading to voice assistants that better understand and respond to natural language. This evolution promises a more seamless and immersive user experience, contributing to the creation of a cohesive and harmonious digital environment.

The project is positioned as a testament to the boundless potential of Python and Flutter, showcasing their adaptability and innovation in meeting the diverse needs of users. It serves as an illustration of how modern technology can address the dynamic requirements of users and adapt to the ever-changing realm of digital applications. The project exemplifies the adaptability and innovation of modern technology to address the diverse needs of users and the ever-evolving landscape of digital applications.

In essence, the conclusion paints a picture of an exciting future where interactive voice assistants, driven by advanced machine learning capabilities, play a central role in shaping a more intuitive, efficient, and user-centric computing experience.

REFERENCES

1) J. Weizenbaum, "ELIZA-a computer program for the study of natural language communication between man and machine," Commumications of the ACM, vol. 9, no. I, pp. 36-45, 1966.

2) K. M. Colby, Artificial Paranoia: A Computer Simulation of Paranoid Process, Pergamon Press. Oxford, UK, 1975

3) R. Dale, "The return of the chatbots," Natural Language Engineering, vol. 22, no. 5, pp. 811-817, 2016

4) Mittal, A. Agrawal, A. Chouksey, R. Shriwas, and S. Agrawal, "A comparative study of chatbots and humans," Situations, vol. 2. p. 2, 2016.

5) D. A. Ferrucci, "Introduction to "this is watson"," IBM Journal of Research and Development, vol. 56, no. 3.4, pp. I-15, 2012.

6) S. Ghose and J. J. Barua, "Toward the implementation of a topic specific dialogue based natural language chatbot as an undergraduate advisor," in Proceedings of the 2013 International Conference on Informatics, Elecetronics and Vision (ICIEV), pp. I-5, IEEE, Dhaka, Bangladesh,May 2013.

7) Eairanti. "Text-only chatbots" VS. "voice-enabled chatbots". Which is better? https://medium.com/the-social-chai/text-only-chatbots-vs-voice-enabled-chatbots-which-is-better-c796e117678f

8) H. Blog, "Voice or Text: Deciding and Designing a User-First Chatbot,"U.D. a. H. Jagrat, Ed., ed.

9) J. P. Mclntire, L. K. Mclntire, and P. R. Havig, "Methods for chatbot detection in distributed text-based communications," in Proceedings of the 2010 International

ADDITIONAL REFERENCES

1. B. A. Abu Shawar, "A corpus based approach to generalising a chatbot system," Univesity of Leeds, Leceds UK, 2005, Ph. D. Thesis.

2. W. Nwankwo, "Interactive advising with bots: improving academic excellence in educational establishments," American Journal of Operations Management and information Systems, vol. 3, no. 1, pp. 6-11, 2018.

3. A. Palanica, P. Flaschner, A. Thommandram, M. Li, and Y. Fossat, "Physicians' perceptions of chatbots in health care: cross-sectional web-based survey." Journal of Medical Internet Research, vol. 21, no. 4, p. el2887, 2019, in English.

4. T. Anh, Artificial Intelligence in E-Comnerce: Case Amazon, Centria University of Applied Sciences, Kokkola, Finland, 2019.

5. R. Khandale, S. Sombansi, S. Mishra, M. F. Shaikh, and P. Mishra, "E-negotiator chatbot for e- commerce websites: implementation," Journal of Applied Science and Computations, vol. 6, no. 6, p. 489, 2019.

6. S. Angelov and M. Lazarova, "E-commerce distributed chatbot system," in Proceedings of the 9th Balkan Conference on lnformatics, p. 8, Sofia, Bulgaria, September 2019.

7. S. Gupta, D. Borkar, C. De Mello, and S. Patil, "An e-commerce website based chatbot," International Journal of Compter Science amd Information Techrologies, vol. 6, no. 2, pp. 1483-1485, 2015.

8. T. Böger., "Implementation and evaluation of a shopping assistance chatbot in an e-commerce case," University of Ljubljana, Ljubljana, Slovenia, 2019, M. Sc. Thesis.

9. S. Reshmi and K. Balakrishnan, "Empowering chatbots with business intelligence by big data 'integration," International Journal of Advanced Research in Computer Science, vol. 9, no. 1, pp. 627-631,2018.

APPENDIX-A
PSUEDOCODE

```
import speech_recognition as sr
import pyttsx3
import wikipedia
import datetime
import webbrowser
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from ecapture import ecapture as ec
from bs4 import BeautifulSoup
from urllib.request import urlopen
from GoogleNews import GoogleNews

def voice():
    speak("Would you prefer a male or a female voice?")
    v = takeCommand()
    v = v.strip()
    v = v.lower()
    if v=='male':
        engine.setProperty('voice', voices[0].id)
    elif v=='female':
        engine.setProperty('voice', voices[1].id)


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

def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour >= 0 and hour < 12:
        speak("Good Morning Sir !")

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

    else:
        speak("Good Evening Sir !")

    vsname = ("Kronos 0 point 4")
    speak("I am your Assistant")
    speak(vsname)
    speak("How may i help you?")


def summary():
```

```
    speak("What should I search for?")
    inp = takeCommand()
    speak("How many sentences do you need?")
    num = takeCommand()
    speak(wikipedia.summary(inp,num))


def mail():
    smtp_server = "smtp.office365.com"
    smtp_port = 587
    speak("Your email address please")
    s = takeCommand()
    if 'dot' in s or 'at' in s or 'at the rate' in s :
        a = s.replace('dot','.')
        z = a.replace('at','@')
        m = z.replace('at the rate','@')
        sender_email = m.replace(" ","")
    print(sender_email)
    speak('is the email address correct? say confirm to proceed')
    con = takeCommand()
    while con!='confirm':
        speak("Your email address please")
        s = takeCommand()
        if 'dot' in s or 'at' in s or 'at the rate' in s:
            a = s.replace('dot','.')
            z = a.replace('at','@')
            m = z.replace('at the rate','@')
            sender_email = m.replace(" ","")
        print(sender_email)
        speak('is the email address correct? say confirm to proceed')
        con = takeCommand()

    speak("Please type your password")
    sender_password = str(input("Enter your password:\n"))
    speak("The recipient's email address please")
    r = takeCommand()
    if 'dot' in r or 'at' in r or 'at the rate' in r:
        a = r.replace('dot','.')
        z = a.replace('at','@')
        m = z.replace('at the rate','@')
        recipient_email= m.replace(" ","")
    print(recipient_email)
    speak('Is the email address correct? say confirm to proceed')
    firm = takeCommand()
    while firm!='confirm':
        speak("Recipient email address please")
        r = takeCommand()
        if 'dot' in r or 'at' in r or 'at the rate' in r:
```

```
            a = s.replace('dot','.')
            z = a.replace('at','@')
            m = z.replace('at the rate','@')
            recipient_email = m.replace(" ","")
        print(recipient_email)
        speak('is the email address correct? say confirm to proceed')
        firm = takeCommand()
    msg = MIMEMultipart()
    msg["From"] = sender_email
    msg["To"] = recipient_email
    speak("What is the subject of the mail")
    msg["Subject"] = takeCommand()
    print(msg["Subject"])
    speak("Is the subject correct? Say confirm to proceed")
    aff = takeCommand()
    while aff!="confirm":
        speak("What is the subject of the mail")
        msg["Subject"] = takeCommand()
        print(msg["Subject"])
        speak("Is the subject correct? Say confirm to proceed")
        aff = takeCommand()
    speak("What is the body of the mail?")
    body = takeCommand()
    print(body)
    speak("Is the body correct? Say confirm to proceed")
    aff1 = takeCommand()
    while aff1 != "confirm":
        speak("What is the body of the mail?")
        body = takeCommand()
        print(body)
        speak("Is the body correct? Say confirm to proceed")
        aff1 = takeCommand()

    msg.attach(MIMEText(body, "plain"))



if __name__ == '__main__':
    clear = lambda: os.system('cls')
    clear()
    voice()
    username()
    wishMe()
    while True:
        strt = takeCommand().lower()
        if 'hey kronos' in strt or 'hey coronavirus' in strt:
            speak(random.choice(start_text))
            query = takeCommand().lower()
```

```
            if 'wikipedia' in query or 'summarize' in query or 'summary' in
query or 'summarise' in query:
                summary()

            elif 'open youtube' in query:
                speak("Here you go to Youtube\n")
                webbrowser.open("https://www.youtube.com")

            elif 'open google' in query:
                speak("Here you go to Google\n")
                webbrowser.open("https://www.google.com")

            elif 'search' in query:
                speak("What should i search for?")
                s = takeCommand()
                speak("Alright, se   arching for '"+ s +"' on google")
                pywhatkit.search(s)

            elif 'play music' in query or "play song" in query:
                speak("Here you go with music")
                music_dir = "C:\\Music"
                songs = os.listdir(music_dir)
                print(songs)
                random_song = os.path.join(music_dir, random.choice(songs))
                os.startfile(random_song)

            elif "what's the time" in query:
                strTime = datetime.datetime.now().strftime("%H:%M:%S")
                speak(f"The time is {strTime}")

            elif 'send a mail' in query or 'mail' in query:
                mail()

            elif 'exit' in query or 'bye' in query or 'goodbye' in query:
                speak("I hope to see you again , goodbye")
                exit()
```

APPENDIX-B

SCREENSHOTS

Main.dart

kronos_inital > lib > main.dart > ...

Click here to ask Blackbox to help you code faster

```dart
import 'package:flutter/material.dart';
import 'package:kronos_underconstruction/screens/first_screen.dart';

Run | Debug | Profile
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Kronos Under Construction',
      theme: ThemeData(
      primarySwatch: Colors.blueGrey,
      ), // ThemeData
      home: FirstScreen(),
    ); // MaterialApp
  }
}

```

Firstscreen.dart

Click here to ask Blackbox to help you code faster

```dart
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:kronos_underconstruction/screens/second_screen.dart';

class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black26,
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Image.asset('assets/images/kronusimage.png', height: 300, width: 300),
                SizedBox(width: 0),
                Text('ronos', style: TextStyle(fontSize: 120, color: Colors.white)),
              ],
            ), // Row
            SizedBox(height: 150),
            ElevatedButton(
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => SecondScreen()),
                );
              },
```

```
29                  },
30                  child: Text('Let\'s Start'),
31                ), // ElevatedButton
32              ],
33            ), // Column
34          ), // Center
35        ); // Scaffold
36      }
37    }
38
```

Chatconsole.dart

```
Click here to ask Blackbox to help you code faster
1   import 'dart:convert';
2   import 'package:flutter/material.dart';
3   import 'package:http/http.dart' as http;
4
5   class ChatConsole extends StatefulWidget {
6     @override
7     _ChatConsoleState createState() => _ChatConsoleState();
8   }
9
10  class _ChatConsoleState extends State<ChatConsole> {
11    List<String> messages = []; // List to store chat messages
12    TextEditingController messageController = TextEditingController();
13
14    @override
15    Widget build(BuildContext context) {
16      return Scaffold(
17        appBar: AppBar(
18          title: Text('Chat Console'),
19        ), // AppBar
20        body: Column(
21          children: [
22            Expanded(
23              child: Container(
24                padding: EdgeInsets.all(8.0),
25                color: Colors.grey.shade200,
26                child: ListView.builder(
27                  itemCount: messages.length,
28                  itemBuilder: (context, index) {
29                    return ListTile(
```

```
                        title: Text(
                          messages[index],
                          style: TextStyle(fontSize: 16.0),
                        ), // Text
                      ); // ListTile
                    },
                  ), // ListView.builder
                ), // Container
              ), // Expanded
              Container(
                color: Colors.white,
                child: Row(
                  children: [
                    IconButton(
                      icon: Icon(Icons.arrow_back),
                      onPressed: () {
                        Navigator.pop(context);
                      },
                    ), // IconButton
                    IconButton(
                      icon: Image.asset('assets/images/mic-dark.png', height: 24, width: 24),
                      onPressed: () {
                        // Handle mic button press
                      },
                    ), // IconButton
                    Expanded(
                      child: TextField(
                        controller: messageController,
                        decoration: InputDecoration(
```

```
                    IconButton(
                      icon: Image.asset('assets/images/send.png', height: 24, width: 24),
                      onPressed: () {
                        sendMessage();
                      },
                    ), // IconButton
                  ],
                ), // Row
              ), // Container
            ],
          ), // Column
        ); // Scaffold
      }

      void sendMessage() async {
        String userMessage = messageController.text;
        if (userMessage.isNotEmpty) {
          setState(() {
            messages.add('You: $userMessage');
          });

          String apiUrl = 'http://127.0.0.1:5000/get-data';
          Map<String, String> headers = {'Content-Type': 'application/json'};
          Map<String, String> body = {'userMessage': userMessage};

          try {
            final response = await http.post(Uri.parse(apiUrl), headers: headers, body: jsonEncode(body));

            if (response.statusCode == 200) {
```

```
92                  Map<String, dynamic> data = json.decode(response.body);
93                  String assistantResponse = data['assistantResponse'];
94
95                  setState(() {
96                    messages.add('Assistant: $assistantResponse');
97                  });
98                } else {
99                  print('Error: ${response.statusCode}');
100               }
101             } catch (e) {
102               print('Error: $e');
103             }
104
105             messageController.clear();
106           }
107         }
108       }
109
          Run | Debug | Profile
110       void main() {
111         runApp(MaterialApp(
112           home: ChatConsole(),
113         )); // MaterialApp
114       }
115
```

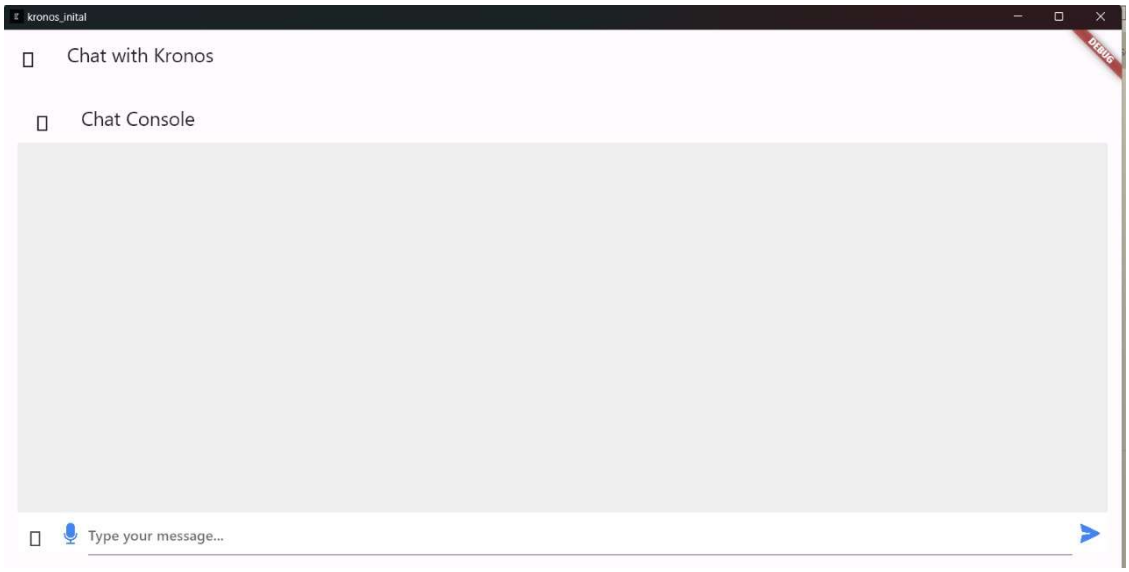
Pubspec.yaml

kronos_inital > pubspec.yaml

Click here to ask Blackbox to help you code faster

```
1   name: kronos_underconstruction
2   description: A Flutter project for Kronos under construction.
3
4   environment:
5     sdk: '^3.2.0'
6
7   dependencies:
8     flutter:
9       sdk: flutter
10    flutter_bloc: ^8.1.0
11    http: ^1.1.0
12    material_color_utilities: 0.5.0
13
14
15
16  flutter:
17    assets:
18    - assets/images/kronusimage.png
19    - assets/images/send.png
20    - assets/images/mic-dark.png
21
22
```

Output (Front-end)



Backend (Python):

```
import subprocess
import pyttsx3
import tkinter
import json
import random
import operator
import speech_recognition as sr
import datetime
import wikipedia
import webbrowser
import os
import winshell
import pyjokes
import feedparser
import PyPDF2
import smtplib
import pywhatkit
import ctypes
import clients
import time
import requests
import wikipedia
from progress.bar import Bar
import shutil
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from ecapture import ecapture as ec
from bs4 import BeautifulSoup
import win32com.client as wincl
from urllib.request import urlopen
from GoogleNews import GoogleNews
```

```
def voice():
    speak("Would you prefer a male or a female voice?")
    v = takeCommand()
    v = v.strip()
    v = v.lower()
    if v=='male':
        engine.setProperty('voice', voices[0].id)
    elif v=='female':
        engine.setProperty('voice', voices[1].id)


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

def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour >= 0 and hour < 12:
        speak("Good Morning Sir !")

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

    else:
        speak("Good Evening Sir !")

    vsname = ("Kronos 0 point 4")
    speak("I am your Assistant")
    speak(vsname)
    speak("How may i help you?")


def summary():
    speak("What should I search for?")
    inp = takeCommand()
    speak("How many sentences do you need?")
    num = takeCommand()
    speak(wikipedia.summary(inp,num))
```

```
def mail():
    smtp_server = "smtp.office365.com"
    smtp_port = 587
    speak("Your email address please")
    s = takeCommand()
    if 'dot' in s or 'at' in s or 'at the rate' in s :
        a = s.replace('dot','.')
        z = a.replace('at','@')
        m = z.replace('at the rate','@')
        sender_email = m.replace(" ","")
    print(sender_email)
    speak('is the email address correct? say confirm to proceed')
    con = takeCommand()
    while con!='confirm':
        speak("Your email address please")
        s = takeCommand()
        if 'dot' in s or 'at' in s or 'at the rate' in s:
            a = s.replace('dot','.')
            z = a.replace('at','@')
            m = z.replace('at the rate','@')
            sender_email = m.replace(" ","")
        print(sender_email)
        speak('is the email address correct? say confirm to proceed')
        con = takeCommand()

    speak("Please type your password")
    sender_password = str(input("Enter your password:\n"))
    speak("The recipient's email address please")
    r = takeCommand()
    if 'dot' in r or 'at' in r or 'at the rate' in r:
        a = r.replace('dot','.')
        z = a.replace('at','@')
        m = z.replace('at the rate','@')
        recipient_email= m.replace(" ","")
    print(recipient_email)
    speak('Is the email address correct? say confirm to proceed')
    firm = takeCommand()
```

```
    while firm!='confirm':
        speak("Recipient email address please")
        r = takeCommand()
        if 'dot' in r or 'at' in r or 'at the rate' in r:
            a = s.replace('dot','.')
            z = a.replace('at','@')
            m = z.replace('at the rate','@')
            recipient_email = m.replace(" ","")
        print(recipient_email)
        speak('is the email address correct? say confirm to proceed')
        firm = takeCommand()
    msg = MIMEMultipart()
    msg["From"] = sender_email
    msg["To"] = recipient_email
    speak("What is the subject of the mail")
    msg["Subject"] = takeCommand()
    print(msg["Subject"])
    speak("Is the subject correct? Say confirm to proceed")
    aff = takeCommand()
    while aff!="confirm":
        speak("What is the subject of the mail")
        msg["Subject"] = takeCommand()
        print(msg["Subject"])
        speak("Is the subject correct? Say confirm to proceed")
        aff = takeCommand()
    speak("What is the body of the mail?")
    body = takeCommand()
    print(body)
    speak("Is the body correct? Say confirm to proceed")
    aff1 = takeCommand()
    while aff1 != "confirm":
        speak("What is the body of the mail?")
        body = takeCommand()
        print(body)
        speak("Is the body correct? Say confirm to proceed")
        aff1 = takeCommand()

    msg.attach(MIMEText(body, "plain"))
```

```
if __name__ == '__main__':
    clear = lambda: os.system('cls')
    clear()
    voice()
    username()
    wishMe()
    while True:
        strt = takeCommand().lower()
        if 'hey kronos' in strt or 'hey coronavirus' in strt:
            speak(random.choice(start_text))
            query = takeCommand().lower()
            if 'wikipedia' in query or 'summarize' in query or 'summary' in query or 'summarise' in query:
                summary()

            elif 'open youtube' in query:
                speak("Here you go to Youtube\n")
                webbrowser.open("https://www.youtube.com")

            elif 'open google' in query:
                speak("Here you go to Google\n")
                webbrowser.open("https://www.google.com")

            elif 'search' in query:
                speak("What should i search for?")
                s = takeCommand()
                speak("Alright, se   arching for '"+ s +"' on google")
                pywhatkit.search(s)

            elif 'play music' in query or "play song" in query:
                speak("Here you go with music")
                music_dir = "C:\\Music"
                songs = os.listdir(music_dir)
                print(songs)
                random_song = os.path.join(music_dir, random.choice(songs))
                os.startfile(random_song)

            elif "what's the time" in query:
                strTime = datetime.datetime.now().strftime("%H:%M:%S")
```

APPENDIX-C

ENCLOSURES

1. Conference Paper Presented Certificate

ISSN: 2582-3930

ACCEPTANCE CERTIFICATE

Impact Factor: 8.176
DOI Prefix: 10.55041

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT (IJSREM)

An Open Access Scholarly Journal || Index in major Databases & Metadata

We are pleased to inform you that your Research Paper titled

Interactive Voice Assistant using Machine Learning Techniques

has been ACCEPTED for publication in

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT (IJSREM)

VOLUME 08 ISSUE 01 JANUARY - 2024

We are delighted to see your commitment & hardwork to share your research is being recognized. We look foreward to helping you with all of your publication needs. Thank you for choosing IJSREM!!

Editor-in-chief
IJSREM

www.ijsrem.com

ijsremjournal@gmail.com

2. Plagiarism Report

REVISED-UP-II PROJECT REPORT TEMPLATE_UPDATED parkavi

ORIGINALITY REPORT

18%	15%	11%	15%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Nanyang Technological University, Singapore Student Paper	3%
2	Submitted to Presidency University Student Paper	2%
3	www.hindawi.com Internet Source	1%
4	downloads.hindawi.com Internet Source	1%
5	Submitted to M S Ramaiah University of Applied Sciences Student Paper	1%
6	P. William, Govinda Rajulu Lanke, Venkata Narasimha Rao Inukollu, Prabhdeep Singh, Anurag Shrivastava, Rohit Kumar. "Framework for Design and Implementation of Chat Support System using Natural Language Processing", 2023 4th International Conference on Intelligent Engineering and Management (ICIEM), 2023 Publication	1%

3. Sustainable Development Goals (SDG):

- The Sustainable Development Goals (SDGs) are a set of 17 global objectives established by the United Nations in 2015 to address various social, economic, and environmental challenges. Designed to be achieved by 2030, the SDGs aim to eradicate poverty, ensure equality, promote sustainable economic growth, and address climate change, among other critical issues. These goals provide a comprehensive framework for international cooperation to create a more inclusive, resilient, and sustainable future for all.
- Our project maps primarily to 4 goals, they are:
    - Goal 4: Quality Education
    - Goal 9: Industry, Innovation, and Infrastructure
    - Goal 10: Reduced Inequalities
    - Goal 13: Climate Action

Goal 4: Quality Education - The code promotes quality education by integrating speech recognition and synthesis, enabling user interaction and learning through voice-based commands.

Goal 9: Industry, Innovation, and Infrastructure - The code showcases innovation by incorporating various libraries and APIs to create a versatile AI assistant, contributing to advancements in technology infrastructure.

Goal 10: Reduced Inequalities - The code aims to reduce inequalities by providing an accessible voice interface, making it inclusive for users with varying levels of technological proficiency.

Goal 13: Climate Action - Although indirectly related, the code supports climate action by being an application that can potentially reduce the need for physical interactions with devices, promoting energy efficiency and reducing carbon footprint.