

VOICE BASED RECOGNITION SYSTEM USING DEEP LEARNING

Submitted in partial fulfillment of the requirement towards award of
Bachelor of Engineering degree in Computer Science and Engineering

By

POOJA R A
(Reg No - 40110962)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

CATEGORY -1 UNIVERSITY BY UGC
Accredited with Grade "A++" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

APRIL – 2024

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Pooja R A (40110962)** who carried out the Project entitled "**VOICE BASED RECOGNITION SYSTEM USING MACHINE LEARNING**" under my supervision from November 2023 to April 2024.

Jany
12/4/24
Internal Guide

Dr. S. L. JANY SHABU, M.Tech., Ph.D.

[Signature]
Head of the Department

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeppiaar Nagar, Rajarajeshwari Salai,
Chennai - 600 079.

Submitted for Viva voce Examination held on 15-04-2024

[Signature]
15/4/24
Internal Examiner

[Signature]
15/4/2024
External Examiner

DECLARATION

I, **Pooja R A (40110962)** hereby declare that the Project Report entitled **Voice Based Recognition System Using Machine Learning** done by me under the guidance of **Dr.S.L.JANY SHABU, M.Tech., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 15/04/2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr.L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. L. Jany Shabu M.Tech, Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

From the past few years, Artificial Intelligence (AI) has shown remarkable progress, and its future is growing day by day. Natural Language Processing (NLP) is an application of AI. A voice assistant utilizes cloud computing to integrate AI and communicate with users in natural language. In households today, millions of devices use voice assistants due to their ease of use. Universities and schools are just starting to use smart speakers with voice assistants. Voice assistants are the greatest innovation in AI that can change people's lives in a significant way. Initially, voice assistants were introduced on smartphones, then they became popular. The idea was widely accepted by all. A voice assistant was initially used in smartphones and laptops, but now it is also found in home automation systems and smart speakers. As devices become smarter, they are interfacing with humans in a language that is easier to understand. Programs based upon desktop voice assistants recognize and respond to human voices via an integrated voice system. We will discuss the main problems and limitations of voice assistants in this paper. The past few years have seen a spate of cyberattacks targeting multinational companies, governments, institutions and even individuals' data. Most of these attacks target the vulnerabilities of the centralized system of data storage, emphasizing that a novel approach to data storage must be developed to tackle long- standing issues with centralized systems. Blockchain technology - a decentralized ledger based system provides a viable alternative to many of these issues. The objective of this project is to develop and test a decentralized document storage system using a private Blockchain network, Ethereum coins and the Inter Planetary File System(IPFS). Interim results suggested that the proposed solution can be implemented at scale for certain use cases including confidential data of both private and public entities.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	1
	1.1 Background And Motivation	2
	1.2 Problem Statement	2
	1.3 Objective And Scope	3
	1.4 Applications Of Voice Assistance	4
2	LITERATURE SURVEY	6
	2.1 Overview of Inference from Research and Literature Survey	6
	2.2 Open problems in Existing System	8
	2.2.1 Data Security	8
	2.2.2 Emotion Recognition	9
	2.2.3 Ethical and Bias Concerns	9
3	REQUIREMENTS ANALYSIS	10
	3.1 Feasibility Studies/Risk Analysis of the Project	10
	3.2 Software Requirements Specification Document	11
	3.2.1 Hardware Requirements	11
	3.2.2 Software Requirements	11
	3.2.3 Python Features	11
4	DESCRIPTION OF PROPOSED SYSTEM	17
	4.1 Selected Methodology or process model	17
	4.2 Architecture / Overall Design of Proposed System	18
	4.2.1 Flow Chart Explanation For Voice Recognition	19
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	25
	4.4 Project Management Plan	27
	4.5 Software To Operation Plan	29

5	IMPLEMENTATION DETAILS	32
	5.1 Development and Deployment System	32
6	RESULT AND DISCUSSION	34
	6.1 Output	34
7	CONCLUSION	36
	7.1 Conclusion	36
	7.2 Recommendations For Future Research	36
	7.3 Research Issues	38
	7.4 Implementation Issues	40
	REFERENCES	42
	APPENDIX	43
	A. Source Code	43
	B. Screen Shot	65
	C. Research Paper	67

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
3.1	Python Programming Language	11
3.2	Visual Studio Code IDE	12
3.3	Ethereum Block Chain	14
4.1	Flow chart of VRS	18
4.2	System Architecture	21
4.3	Architecture of distributed storage system based on Blockchain	24
B1	Voice Command System Output	65
B2	Application Opening Output	65
B3	Information Gathering System	66
B3	Decentralized File Storage System	73

LIST OF ABBREVIATIONS

ABBREVIATIONS

API

VRS

PVA

NFT

ETH

EXPANSIONS

Application Programming Interface

Voice Recognition System

Personal Voice Assistant

Non Fungible Tokens

Ethereum

CHAPTER 1

INTRODUCTION

Artificial Intelligence when used with machines, it shows us the capability of thinking like humans. In this, a computer system is designed in such a way that typically requires interaction from human. As we know Python is an emerging language so it becomes easy to write a script for Voice Assistant in Python. The instructions for the assistant can be handled as per the requirement of user. Speech recognition is the Alexa, Siri, etc. In Python there is an API called Speech Recognition which allows us to convert speech into text. It was an interesting task to make my own assistant. It became easier to send emails without typing any word, searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. Speech is an effective and natural way for people to interact with applications, complementing or even replacing the use of mice, keyboards, controllers, and gestures. A hands-free, yet accurate way to communicate with applications, speech lets people be productive and stay informed in a variety of situations where other interfaces will not. Speech recognition is a topic that is very useful in many applications and environments in our daily life. Generally speech recognizer is a machine which understands humans and their spoken word in some way and can act thereafter. A different aspect of speech recognition is to facilitate for people with functional disability or other kinds of handicap. To make their daily chores easier, voice control could be helpful. In recent years, cloud storage systems have been widely used to meet demands for companies, organizations, and individuals in terms of data storage and sharing. Users investing in these kinds of services must fully put their trust in those companies to keep their important and private data secured for a long period of time. However, centralized systems tend to pose a threat of being attacked or services being interrupted heavily. Moreover, information leakage cases such as Facebook Cambridge Analytica, have led to a strong movement from using centralized to decentralized data storage systems at the time.

1.1 BACKGROUND AND MOTIVATION

Accessibility: Not everyone is proficient in typing or reading text, but almost everyone can speak. A voice-based reporting system opens up reporting capabilities to a broader audience, including those with disabilities or those who are illiterate. This inclusivity aligns with principles of equity and ensures that everyone has the opportunity to participate in reporting processes.

Efficiency: Speaking is often faster than typing, especially for individuals who are accustomed to verbal communication. With a voice-based system, users can dictate their reports in real-time, potentially saving significant time compared to typing out the same information. This efficiency can be particularly valuable in time-sensitive situations where rapid reporting is essential, such as emergency response scenarios or live event coverage.

Reduced Workload: For professionals who need to document their activities or observations as part of their work, such as field researchers, journalists, or healthcare providers, voice-based reporting can streamline the documentation process. Rather than spending time transcribing handwritten or voice-recorded notes, users can directly input their reports through speech, reducing the administrative burden and freeing up time for other tasks.

1.2 PROBLEM STATEMENT

The problem statement in voice recognition systems typically revolves around improving accuracy, enhancing robustness across diverse environments, and advancing the usability and functionality of the systems.

Accuracy Improvement: Despite significant advancements in voice recognition technology, accuracy remains a persistent challenge, particularly in noisy environments, with accents, or for users with speech impairments. The problem statement often includes the need to enhance the accuracy of voice recognition systems across various contexts and user demographics.

Privacy and Security: Voice recognition systems raise concerns about privacy and security, particularly regarding the collection, storage, and usage of users' voice data. The problem statement encompasses developing robust privacy-preserving techniques, ensuring secure transmission and storage of voice data, and implementing user-friendly mechanisms for obtaining consent and managing data access.

1.3 OBJECTIVE AND SCOPE

Objective

The objective of the voice recognition system using machine learning is to accurately identify and transcribe spoken words or commands into text format. This system aims to enhance user experience by providing seamless interaction with devices and applications through natural language processing. By leveraging machine learning techniques, the system seeks to improve accuracy, adaptability, and robustness in recognizing various accents, languages, and speech patterns. The primary objective of implementing a voice recognition system using machine learning is to revolutionize human-computer interaction by enabling seamless communication through spoken language. By harnessing the power of machine learning algorithms, this system aims to accurately transcribe spoken words into text format, facilitating natural and intuitive interactions with devices and applications. Through precise speech recognition capabilities, the system endeavors to enhance accessibility for individuals with disabilities, as well as improve overall user experience for all users. Moreover, the objective extends beyond mere transcription to encompass speaker identification, allowing for personalized interactions tailored to individual users. Additionally, the system seeks to optimize efficiency and reliability by continually learning from user interactions, thereby improving its accuracy and adaptability over time. Overall, the objective of this voice recognition system is to redefine the way humans interact with technology, making communication more intuitive, efficient, and inclusive.

Scope

The scope of the voice recognition system using machine learning encompasses various aspects crucial for its functionality and effectiveness. It involves developing algorithms for accurate speech recognition across languages and dialects, implementing speaker identification for personalized interactions, and enabling keyword spotting for command-based interactions. Additionally, the scope includes addressing challenges such as noise reduction, real-time processing, and adaptability through continuous learning. Integration with diverse applications and platforms, ensuring accessibility and security, and scalability are also integral parts of the scope. By focusing on these aspects, the system aims to provide a seamless and efficient user experience while interacting with technology through voice commands. Beyond the fundamental aspects of accurate speech recognition and speaker identification, the system aims to integrate advanced features such as contextual understanding, multimodal interaction, and emotion recognition. Furthermore, customization options, continuous improvement mechanisms, and cross-platform compatibility are essential components to ensure the system's adaptability and user satisfaction. Additionally, industry-specific applications, multilingual support, and compliance considerations add further dimensions to its scope, catering to diverse user needs and regulatory requirements. By embracing these various facets, the voice recognition system seeks to provide a comprehensive and versatile solution for natural language interaction with technology, enhancing user experience and accessibility across different domains and contexts.

1.4 APPLICATIONS OF VOICE ASSISTANCE

Virtual Assistants: Virtual assistants like Amazon Alexa, Apple's Siri, Google Assistant, and Microsoft Cortana utilize voice recognition technology to understand and respond to user commands and queries. They can perform tasks such as setting

reminders, providing weather updates, controlling smart home devices, and searching the internet, all through voice interaction.

Speech-to-Text Transcription: Voice recognition systems are used for converting spoken language into text, facilitating tasks such as transcribing interviews, meetings, lectures, and dictation. They find applications in journalism, legal documentation, medical transcription, and general note-taking.

Healthcare Applications: In healthcare, voice recognition systems assist healthcare professionals in documenting patient encounters, creating medical records, and transcribing clinical notes. Voice-enabled electronic health record (EHR) systems streamline documentation workflows, allowing clinicians to focus more on patient care.

Language Translation: Voice recognition technology combined with machine translation enables real-time language translation services. Users can speak in one language, and the system translates their speech into another language, making multilingual communication more accessible and efficient in diverse settings such as travel, business meetings, and international conference

CHAPTER 2

LITERATURE SURVEY

2.1 OVERVIEW OF EXISTING RESEARCH AND LITREATURE SURVEY

Problem statement has been extensively studied over the past decade by researchers and software companies in a bid to create a solution, and all their solutions vary from analyzing various patterns of acoustic data to decentralizing the data storage of the driver.

The work of Saadman Shahid Chowdury et.al [5] on "Domain specific Intelligent personal assistant with bilingual voice command processing." that introduced the approach for creating an Intelligent Personal Assistant which is capable of understanding domain specific direct voice commands from users. By selecting only the subject words and action words for training, the size required for the corpus is smaller. The generalizer and FSA allows the creation of an efficient solution to natural language processing when developers do not have access to sufficient training data.

The study witnessed some major limitations in its execution like the sphinx-4 framework has its own audio noise reduction system, but it is limited in its abilities, and from our experience, attempting to add our own noise cancelling module increases the word error rate of existing acoustic models. The FSA needs to be designed by the developer. hence, when creating an FSA to process more complex commands, the developer needs to ensure that all possible cases are accounted.

The study conducted on "Hey, Siri", "Ok, Google", "Alexa". Acceptance Relevant Factors of Virtual Voice-Assistants by Laura BUR bach et.al [7] on 2019 have shown that virtual voice-assistants are used for manifold purposes. Besides their promising potential, many users are concerned about their privacy and what happens to the data recorded by the voice assistant. It is not yet clear which factors contribute to the acceptance of voice assistants. To address this issue, they conducted a Conjoint Analysis with three attributes and three levels each. Relating the well-known privacy

utility trade-off, we found out that not the price of a voice assistant is the most important factor for its acceptance, but privacy.

Saqib Ali et.al [10] suggested in their research paper that the block chain is a peer-to-peer distributed ledger in which records called blocks are linked and secured using a cryptographic hash. By design, block chains are decentralized, secure, immutable and extremely fault tolerance making them suitable for record management activities i.e., financial transactions, identity management, provenance and authentication. Block chain can be deployed as permission less (e.g., Bitcoin or Ethereum) or permissioned block chain.

Peng Cheng et.al [8] work on “Personal Voice Assistant Security and Privacy—A Survey” suggested that Personal voice assistants (PVAs) are increasingly used as interfaces to digital environments. Voice commands are used to interact with phones, smart homes, or cars. In the United States alone, the number of smart speakers, such as Amazon’s Echo and Google Home, has grown by 78% to 118.5 million, and 21% of the U.S. population own at least one device. Given the increasing dependency of society on PVAs, security and privacy of these have become a major concern of users, manufacturers, and policy makers. Consequently, a steep increase in research efforts addressing security and privacy of PVAs can be observed in recent years. While some security and privacy research applicable to the PVA domain predates their recent increase in popularity, many new research strands have emerged.

R. Zhang et.al [11] cyberattack on smartphones to monitor voice assistant functions in the name of “Using AI to hack IA: A new stealthy spyware against voice assistance function in smartphones”. Pankaj Kuneekar et.al [4] published an extensive research paper on “AI-based Desktop Voice Assistant”. From the past few years, Artificial Intelligence (AI) has shown remarkable progress, and its future is growing day by day. Natural Language Processing (NLP) is an application of AI. A voice assistant utilizes cloud computing to integrate AI and communicate with users in natural language. In households today, millions of devices use voice assistants due to their ease of use. Universities and

schools are just starting to use smart speakers with voice assistants. Voice assistants are the greatest innovation in AI that can change people's lives in a significant way. Initially, voice assistants were introduced on smartphones, then they became popular. The idea was widely accepted by all. A voice assistant was initially used in smartphones and laptops, but now it is also found in home automation systems and smart speakers.

The way AI-powered voice assistants function struck a nerve for Dr. Rand Hindi, a recent Forbes 30 Under 30 entrepreneur. Hindi has been concerned about data privacy for a number of years. He started coding at the early age of 10 and founded his first tech company when he was just fourteen years-old. According to Hindi, leading companies like Google and Amazon are misusing users' personal data collected from AI-based voice assistants like Alexa, Siri and Google Assistant. To ensure that users' data is kept private, Hindi founded Snips, a leading AI company based in Paris and New York that is focused on bringing privacy to AI-based voice technologies. Since 2013, Snips has been building voice technology to guarantee "privacy by design."

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

2.2.1 *Data Security*

Data security is a paramount concern with personal voice assistants. These systems are constantly listening and recording, which raises privacy issues. Here are some key considerations:

- **Data Storage and Retention:** Voice assistant providers must be transparent about how long they store user data and allow users to delete their data if desired.
- **Data Encryption:** All data transmitted between the voice assistant and the cloud servers should be encrypted to prevent interception by malicious actors.
- **Anonymization:** Personal information should be anonymized or pseudo-anonymized to reduce the risk of personal identification.
- **User Consent:** Users should have the ability to opt-in or opt-out of data collection and be aware of what data is being collected and for what purposes.

2.2.2 *Emotion Recognition*

Emotion recognition is a capability some voice assistants are beginning to incorporate. Here are some aspects to consider:

- ***Ethical Use:*** Emotion recognition should be used ethically, and user consent is vital. It has the potential for misuse, such as invasive monitoring.
- ***Accuracy and Bias:*** Ensuring that emotion recognition is accurate across different demographics and doesn't introduce or perpetuate biases is a challenge.
- ***Privacy:*** The voice assistant should recognize emotions based on voice tone and content without violating user privacy by trying to analyze emotions directly.

2.2.3 *Ethical and Bias Concerns*

Ethical and bias concerns are crucial in the development and deployment of voice assistants:

- ***Bias in Training Data:*** Training data used for voice assistants can contain biases. Developers need to actively work on reducing and mitigating these biases.
- ***Diverse Representation:*** Efforts should be made to include diverse voices, backgrounds, and experiences in the development process to ensure inclusivity.
- ***Transparency:*** Users should know how voice assistants work, what data is collected, and how decisions are made to maintain transparency.
- ***Avoiding Harm:*** Developers need to ensure that voice assistants do not promote harmful or discriminatory behavior and that they are designed to prevent engagement in illegal or harmful activities.
- ***Algorithmic Fairness:*** Implementing algorithms that are fair and just, so that voice assistants do not favor one group over another.
- ***User Control:*** Providing users with control over what their voice assistants say or do, allowing them to customize responses within ethical boundaries.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

- ***Concerns about data security***

Despite the fact that people are using voice assistants more frequently, there is still a lot of worry about the information these devices collect and the businesses that create the apps that run on them. Customers are concerned about the data's storage methods, viewers, and eventual disposition. If marketers don't handle these data and privacy issues, they won't be able to access these prospects or their data.

- ***Disconnected exchange***

Another drawback is that compared to other platforms, voice assistants as a channel offer fewer enriching interactions. The choices include visual interactions versus speech material alone, which generally means recycling current content. This might make some of the more significant interactions that marketers can have elsewhere less effective.

- ***Reliance on gadget manufacturers***

You are at the mercy of the manufacturers of devices, such as the corporations that build wearable tech, cars, and appliances, as a marketer. Before getting started, you should carefully investigate the device manufacturers you wish to collaborate with to achieve long-term success. The cost of creating the voice app for this channel can be high. Building an internal skill set tailored towards the intricacies of voice assistants may take a lot of time if you participate in this channel.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

3.2.1 Hardware Requirements

- RAM : 4GB or above 4GB
- Processor of Frequency: 1.5GHz or above
- Processor : Intel

3.2.2 Software Requirements

- Operating System : Window 8 and above
- Languages : Python
- Tools Used : Visual studio code, Ethereum



Fig 3.1: Python Programming Language

Python is a programming language commonly used for developing websites and software, task automation, data analysis, and data visualization. Since it's relatively easy to learn, Python has been adopted by many non-programmers, such as accountants and scientists, for a variety of everyday tasks, like organizing finances.

3.2.3 PYTHON FEATURES

- **Easy-to-learn** - Python has a small number of keywords, a basic structure, and a well-defined syntax. This enables the pupil to swiftly learn the language.
- **Easy-to-read** - Python code is more clearly defined and visible to the eyes.

- **Easy-to-maintain** - Python's source code is relatively simple to maintain.
- **A broad standard library** - The majority of Python's library is portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** - Python features an interactive mode that enables for interactive testing and debugging of code snippets.
- **Portable** - Python can operate on a broad range of hardware devices and has the same interface across them all.
- **Extendable** - The Python interpreter may be extended using low-level modules. These modules allow programmers to enhance or adapt their tools to make them more efficient.
- **Databases** - Python provides interfaces to all major commercial databases.
- **GUI Programming** - Python can construct and port GUI programme to numerous system calls, libraries, and windows systems, including Windows
 - MFC, Macintosh, and Unix's X Window system.
- **Scalable** - Python provides a better structure and support for large program than shell scripting.



Fig 3.2: Visual Studio Code IDE

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality. Cross-Platform Support: Visual Studio Code is available for Windows, macOS, and Linux, providing a consistent development experience across different operating systems.

IntelliSense: IntelliSense is a feature that provides context-aware code completion suggestions, parameter hints, and code navigation capabilities as you type, helping to improve productivity and reduce errors.

Syntax Highlighting: VS Code offers syntax highlighting for a wide range of programming languages, making code easier to read and understand by highlighting different elements such as keywords, variables, and comments with distinct colors.

Code Snippets: Code snippets are predefined templates or shortcuts for common code patterns, allowing developers to quickly insert code blocks and avoid repetitive typing. VS Code supports both built-in and user-defined code snippets.

Integrated Terminal: VS Code includes an integrated terminal that allows developers to run command-line tools, compile code, execute scripts, and perform other tasks without leaving the editor. The terminal can be customized and configured to use different shells and command interpreters.

Version Control Integration: VS Code seamlessly integrates with version control systems such as Git, providing features like source control management, diff viewing, commit history visualization, and conflict resolution directly within the editor.

Debugging Support: Visual Studio Code offers built-in debugging capabilities for various programming languages and platforms. Developers can set breakpoints, inspect variables, step through code execution, and diagnose issues using the integrated debugger.

Extensions Marketplace: VS Code has a rich ecosystem of extensions available through its marketplace, offering additional features, language support, themes, and tools to customize and extend the editor according to specific development needs.

Task Automation: VS Code supports task automation through its built-in task runner system and integration with build tools such as Grunt, Gulp, and npm scripts. Developers can define custom tasks to automate common development tasks like compiling code, running tests, and deploying applications.

Integrated Git UI: In addition to command-line Git integration, VS Code includes a user-friendly Git interface with features like staged changes, commit history, branch management, and conflict resolution, making it easier to work with version-controlled projects.

Extension Development: Visual Studio Code itself is highly extensible, allowing developers to create custom extensions and plugins to add new functionality or integrate with third-party services. The VS Code extension API provides access to various editor features and services for extension development.

Customization and Theming: VS Code offers extensive customization options, including themes, key bindings, editor settings, and UI layouts. Users can personalize their development environment to suit their preferences and workflow preferences.



Fig 3.3: Ethereum Block Chain

Ethereum is a network of computers all over the world that follow a set of rules called the Ethereum protocol. The Ethereum network acts as the foundation for communities, applications, organizations and digital assets that anyone can build and use. Ethereum was conceived in 2013 by programmer Vitalik Buterin. Additional founders of Ethereum included Gavin Wood, Charles Hoskinson, Anthony Di Iorio and Joseph Lubin. In 2014, development work began and was crowdfunded, and the network went live on 30 July

2015. Ethereum allows anyone to deploy permanent and immutable decentralized applications onto it, with which users can interact.

Smart Contracts: Ethereum introduced the concept of smart contracts, which are self-executing contracts with the terms of the agreement directly written into code. Smart contracts automatically enforce the rules and conditions of the agreement, eliminating the need for intermediaries and enabling trustless transactions.

Decentralization: Ethereum is decentralized, meaning it operates on a network of computers (nodes) distributed worldwide, with no single point of control. This decentralization ensures censorship resistance, fault tolerance, and resilience against attacks or failures.

Ether (ETH) Cryptocurrency: Ether (ETH) is the native cryptocurrency of the Ethereum platform. It is used as a medium of exchange for transactions on the network, as well as a reward for miners who validate and secure blocks of transactions through a process called mining.

Consensus Mechanism: Ethereum currently uses a Proof of Work (PoW) consensus mechanism, similar to Bitcoin, where miners compete to solve complex mathematical puzzles to validate transactions and add them to the blockchain. Ethereum is transitioning to a Proof of Stake (PoS) consensus mechanism with the Ethereum 2.0 upgrade, which aims to improve scalability, energy efficiency, and security.

EVM (Ethereum Virtual Machine): The Ethereum Virtual Machine is a runtime environment that executes smart contracts on the Ethereum network. It allows developers to write smart contract code in high-level languages like Solidity and compile it into bytecode that can be executed by the EVM.

Interoperability and Standards: Ethereum supports interoperability with other blockchains and platforms through standards such as ERC (Ethereum Request for Comments) token standards. ERC-20, ERC-721, and ERC-1155 are examples of token standards that define common rules and interfaces for creating fungible tokens, non-fungible tokens (NFTs), and multi-token contracts, respectively.

Decentralized Applications (DApps): Ethereum enables the development and deployment of decentralized applications (DApps) that run on the blockchain. DApps leverage smart contracts to automate processes, manage digital assets, and facilitate peer-to-peer interactions without relying on centralized servers or intermediaries.

Immutable Ledger: The Ethereum blockchain maintains an immutable ledger of all transactions and smart contract interactions since its inception. Once data is recorded on the blockchain, it cannot be altered or deleted, providing a transparent and tamper-proof record of historical transactions.

Scalability Solutions: Ethereum has faced challenges with scalability due to limitations in transaction throughput and network congestion. Various scalability solutions are being developed and implemented, including layer 2 scaling solutions like Ethereum's Optimistic Rollups and Plasma, as well as sharding and off-chain scaling solutions.

Community and Ecosystem: Ethereum has a vibrant and diverse community of developers, entrepreneurs, researchers, and enthusiasts who contribute to its development and adoption. The Ethereum ecosystem encompasses a wide range of projects, tools, libraries, and services built on top of the Ethereum platform, driving innovation and growth in the blockchain space.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

The system executes commands given by the user. Thus, it highly depends on just the voice commands given by the user to complete his job. Designing a voice- controlled computer interface not only makes the execution of a command easier, but it also helps the disabled individuals to control a computer. Hand-free computing is possible where a user can interact with the computer without the use of their hands. Speech recognition can be trained to recognize various voice commands. Disabled persons may find the hand-free computer is vital in their life. This system is designed to recognize the speech and execute with full capability Synthesizing means it converts text to speech. The user is asked to provide voice command by using a microphone. The microphone shall take the speech as a command, and the analogue signals are converted to digital in the internal circuits. Then digital signals are processed as an acoustic model. Once the particular applications are identified by the system, it then opens the application. When the application is found, the system shall prompt the user to create a new application in the current working directory. The system would ask for various operation such as edit, read, paste, copy, and other similar operations that shall be performed inside it once the application opens. The system uses the ferment synthesis, a type of speech synthesizer for responding to the user through voice command.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

Voice assistants are all written in programming languages, which listens the verbal commands and respond according to the user's request. In this project I have used Python Programming language to build the AI-based voice assistant. A user can say,"Play me a song","Open google.com" the voice assistant will respond with the results by playing that particular song or by opening Facebook website. The Voice assistant waits for a pause to know that users have finished their request, then the voice assistant sends users request to its database to search for the request.

The request asked by the user gets split into separate commands, so that our voice assistant can be able to understand. Once within the commands list, our request is searched and compared with the other requests. The commands list then sends these commands back to the Voice assistant. Once the voice assistant receives those commands, then it knows what to do next. The voice assistant would even ask a question if there quest is not clear enough to process it, in other words, to make sure it understands what we would like to receive. If it thinks, it understands enough to process it, the voice assistant will perform the task which the user has asked for. The methodology of the proposed system mainly focuses on development of a voice assistant with the integration of decentralization in respect to achieve data privacy and provide control of the user's data to them. For this all the data collected will be stored and retrieved from ethereum block chain. Some other sort of block chain can also be implemented in the future versions.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

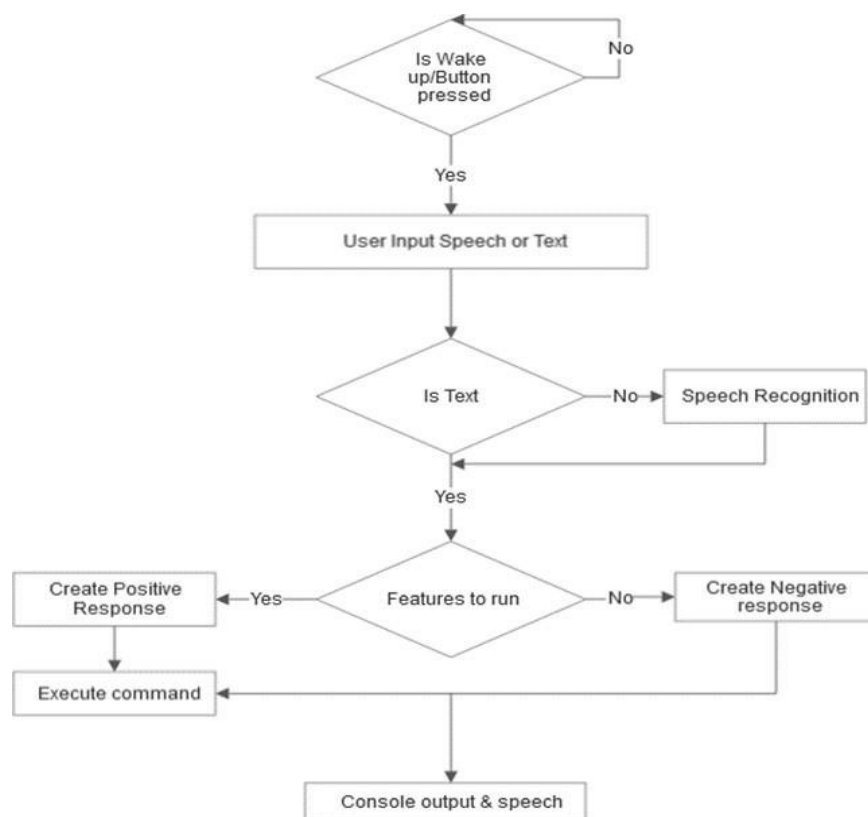


Fig 4.1: Flow chart of VRS

The figure above shows the workflow of the main method of voice assistant. Speech recognition is used to convert speech input to text. This text is then sent to the processor, which determines the character of the command and calls the appropriate script for execution. But that's not the only complexity. No matter how many hours of input, another factor plays a big role in whether a package notices you. Ground noise simply removes the speech recognition device from the target. This may be due to the inability to essentially distinguish between the bark of a dog or the sound near hearing that a helicopter is flying overhead from your voice.

The voice-based recognition system functions by capturing audio input from a microphone, which is then preprocessed to enhance quality and extract relevant features like Mel-frequency cepstral coefficients (MFCCs). These features are analyzed by an acoustic model, typically a deep neural network (DNN), to recognize phonetic units from the speech. Simultaneously, a language model, often an RNN, predicts word sequences based on the recognized phonetic units and context. Through decoding using algorithms like beam search, the system determines the most probable sequence of words. Post-processing steps such as spell checking and semantic analysis refine the recognized text, providing accurate outputs. Finally, the system presents the recognized text or commands to the user, either visually or through synthesized speech. Such a system, exemplified in a virtual assistant application, offers efficient and intuitive interaction through voice commands.

4.2.1 FLOW CHART EXPLANATION FOR VOICE RECOGNITION

Start: This is the starting point of the flowchart, indicating the initiation of the voice recognition process.

Input Voice Command: The user speaks a command or phrase into a microphone or input device.

Audio Recording: The audio input is captured and recorded by the system.

Preprocessing: The recorded audio may undergo preprocessing steps such as noise reduction, normalization, or filtering to improve the quality and clarity of the audio signal.

Feature Extraction: Features or characteristics of the audio signal, such as pitch, frequency, and amplitude, are extracted to represent the spoken words in a format suitable for analysis.

Acoustic Model: The extracted features are compared against an acoustic model, which is a statistical model trained to recognize patterns in speech signals. The acoustic model helps identify phonemes, syllables, or words present in the audio input.

Language Model: In addition to the acoustic model, a language model is used to assess the probability of different word sequences or phrases occurring in the spoken language. The language model helps determine the most likely interpretation of the spoken words based on context and grammar rules.

Decoding: The acoustic and language models work together to decode the audio input and generate a list of possible word hypotheses or transcription candidates.

Candidate Selection: The system selects the most likely transcription candidate or sequence of words based on statistical probabilities and confidence scores calculated from the acoustic and language models.

Post-Processing: The selected transcription candidate may undergo post-processing steps such as error correction, context-aware correction, or smoothing to improve accuracy and readability.

Output Text: The final output of the voice recognition process is the transcribed text representing the spoken words or commands.

Action Execution (Optional): In cases where voice commands are used to control devices or perform actions, the transcribed text may be further processed to execute

the corresponding action or trigger a specific response.

End: This is the endpoint of the flowchart, indicating the completion of the voice recognition process

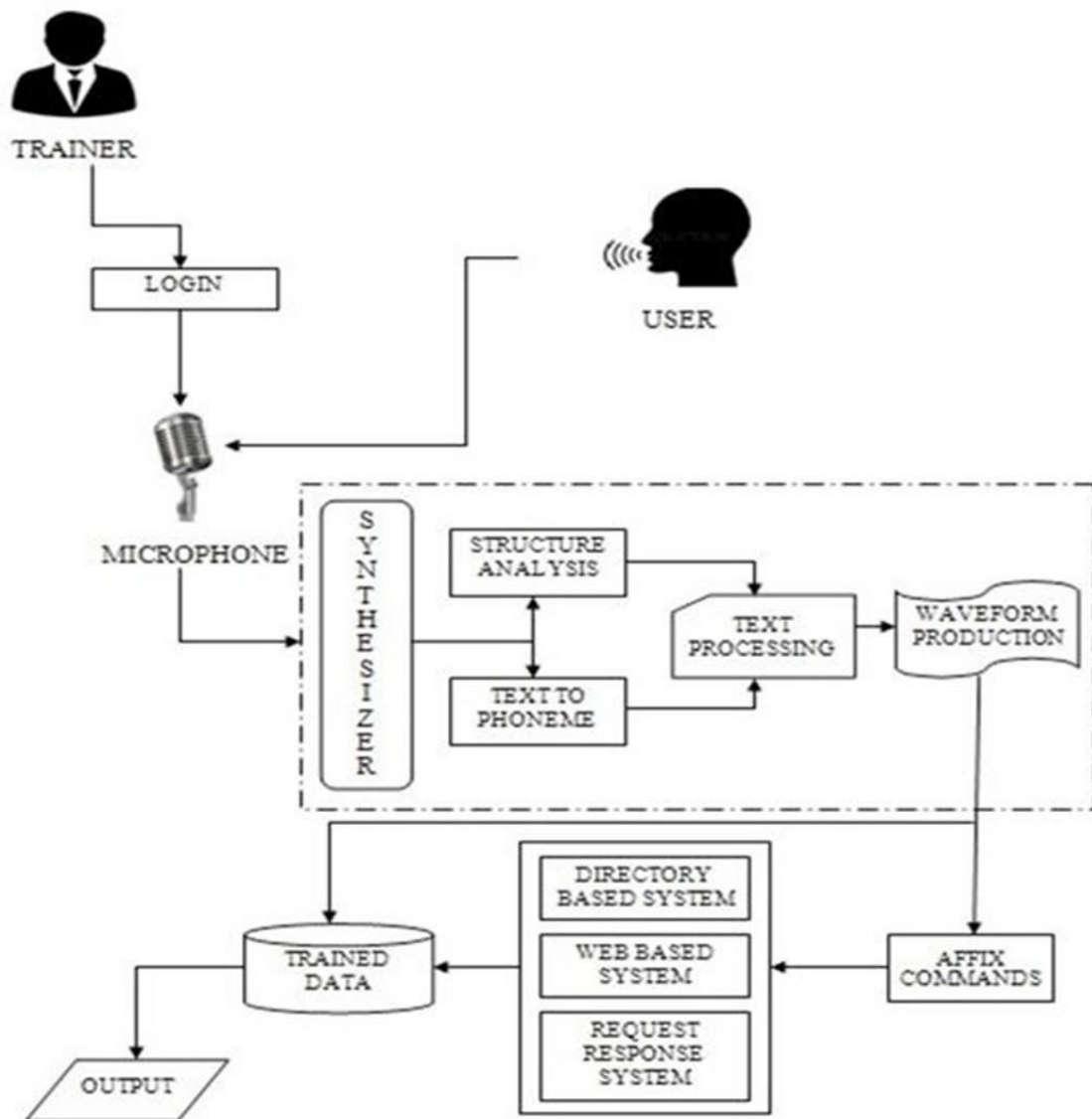


Fig 4.2 :System Architecture of Voice Recognition

This Assistant consists of three modules. First is, assistant accepting voice input from user. Secondly, analyzing the input given by the user, and mapping it to the respective intent and function. And the third is, the assistant giving user the result all along with voice. Initially, the assistant will start accepting the user input. After receiving the input, the assistant will convert the analog voice input to the digital text. A voice recognition system architecture outlines the components and their interactions involved in the process of converting spoken words into text or performing actions based on voice commands. Here's an explanation of the typical architecture of a voice recognition system.

Audio Input: The voice recognition system starts with an audio input, which can be captured using microphones or other audio input devices. The audio input contains the spoken words or commands that need to be recognized.

Preprocessing Module: The raw audio input may undergo preprocessing to enhance its quality and clarity. Preprocessing steps may include noise reduction, normalization, filtering, and segmentation to remove background noise and isolate the speech signal.

Feature Extraction: The preprocessed audio signal is then analyzed to extract relevant features or characteristics that represent the speech content. Common features extracted from the audio signal include Mel-frequency cepstral coefficients (MFCCs), spectrograms, and pitch contours, which capture information about the frequency, amplitude, and temporal characteristics of the speech signal.

Acoustic Model: The extracted features are fed into an acoustic model, which is a statistical model trained to recognize patterns in speech signals. The acoustic model maps the observed speech features to phonemes, syllables, or words and generates hypotheses about the sequence of words present in the audio input.

Language Model: In addition to the acoustic model, a language model is used to assess the probability of different word sequences or phrases occurring in the spoken language. The language model incorporates linguistic knowledge, grammar rules, and contextual information to determine the most likely interpretation of the speech input.

Decoding Module: The acoustic and language models work together in a process called decoding to generate a list of possible word hypotheses or transcription candidates. The decoding module evaluates the compatibility of different word sequences with the observed speech features and selects the most likely transcription candidate based on statistical probabilities and confidence scores.

Post-Processing Module: The selected transcription candidate may undergo post-processing steps to refine the output and improve its accuracy. Post-processing techniques may include error correction, context-aware correction, language model adaptation, and smoothing to ensure the transcribed text is coherent and grammatically correct.

Output Text: The final output of the voice recognition system is the transcribed text representing the spoken words or commands. The transcribed text can be further processed and utilized for various applications, such as natural language understanding, text-to-speech synthesis, or command execution.

Action Execution (Optional): In cases where voice commands are used to control devices or perform actions, the transcribed text may be further processed to execute the corresponding action or trigger a specific response. This step involves interfacing with external systems or applications to carry out the desired actions based on the recognized voice commands.

User Interface (UI): The voice recognition system may include a user interface that provides feedback to the user, such as visual indicators or audio cues, to indicate the status of the recognition process and assistive prompts for interaction.

Overall, the architecture of a voice recognition system encompasses several interconnected modules that work together to analyze, interpret, and transcribe spoken input accurately, enabling seamless interaction with technology through voice commands and speech recognition.

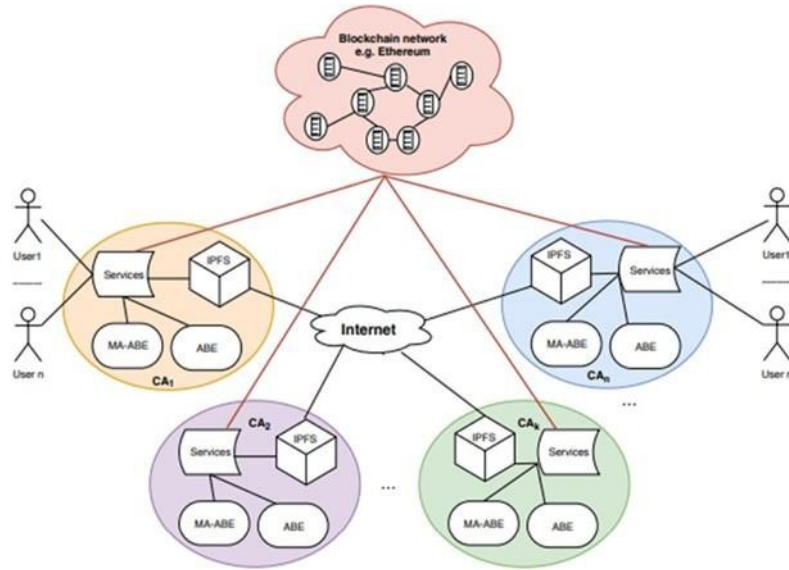


Fig 4.3:Architecture of distributed storage system based on Blockchain

A block chain-based distributed storage system includes a user, an intermediary, and a storage provider. The system flow is mainly composed of Register, Setup, Challenge-proof and Pay phases. The Register phase is the system middleman's access phase to the user and the storage provider. The interaction information is stored in the block chain. The Setup phase is distributed to the system in the early stage. The user encrypts the file data and hands it to the middleman. The middleman encrypts the file. The data block is distributed to the storage provider; the Challenge- proof phase is the storage provider's remote integrity certification phase for storing data, the user issues a data integrity challenge to the storage provider, and the storage provider proves. If it proves successful, enter the Pay phase. If the proof is unsuccessful, the user will no longer pay for the storage.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

Speech Recognition module

Since we're creating a voice assistant app, one of the most critical features is that your assistant knows your voice. In the terminal, execute the following command to install this module.

Date Time module

Date time package is used to showing Date and Time. This date time module comes with built in Python.

Wikipedia

We all know Wikipedia is a great and huge source of knowledge just like Geeks for Geeks or any other sources we have used the Wikipedia module in our project to get more information from Wikipedia or to perform a Wikipedia search. To install this Wikipedia module use pip install Wikipedia.

Web browser

To perform web search. This module comes built-in with Python.

OS

The OS module in Python provides functions for interacting with the os. OS comes under Python's standard utility modules. This module provides a way of using operating system dependent functionality.

PyAudio

PyAudio is a set of Python bindings for Port Audio, a cross- platform C++ library interfacing with audio drivers.

PyQt5

PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android. PyQt5 may also be embedded in C++ based applications to allow users of those applications to configure or enhance the functionality of those applications.

Python Backend

The python backend gets the output from the speech recognition module and then identifies whether the command or the speech output is an API Call and Context Extraction. The output is then sent back to the python backend to give the required output to the user.

Text to speech module

Text-to-Speech (TTS) refers to the ability of computers to read text aloud. A TTS Engine converts written text to a phonemic representation, then converts the phonemic representation to waveforms that can be output as sound. TTS engines with different languages, dialects and specialized vocabularies are available through third-party publishers.

Speech to text conversion

Speech Recognition is used to convert speech input to textual output. It decodes the voice and converts it into a textual format which will be understood by pc easily.

Content Extraction

Context extraction (CE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. In most cases, this activity concerns processing human language texts using natural language processing (NLP). Recent activities in multimedia document processing like automatic annotation and content extraction out of images/audio/video could be seen as context extraction test results.

Textual output

It decodes the voice command and performs the operation then shows the voice command as textual output in the terminal

4.4 PROJECT MANAGEMENT PLAN

Based on the survey we recommend that the application should be developed which accomplishes the desire of different users. The main reason that the user wants to use the voice assistant is to make their life easier, so by implementing the below mentioned features the user can be facilitated.

- Developing for different languages and different accents.
- Portability for any environment.
- Voice authentication technology can be implemented for more security.
- Chatbot implementation requires corpus.
- Dialogue flow needs stack with neurals.
- Deploy on web using flask or Djang.
- Deploy on cloud uses amazon ec2, Heroku.
- NLP features such as finding entities, topic modelli

Project Scope:

- Define the features and functionalities of the voice recognition system.
- Specify any limitations or constraints, such as budget, timeline, and available resources.

Work Breakdown Structure (WBS):

- Break down the project into manageable tasks and sub-tasks.
- Assign resources, dependencies, and timelines to each task.

Schedule Management:

- Develop a project timeline with milestones and deadlines.

- Use project management tools to track progress and adjust schedules as needed.

Resource Management:

- Identify the resources required for the project, including personnel, hardware, and software.
- Allocate resources efficiently to ensure timely completion of tasks.

Risk Management:

- Identify potential risks and uncertainties associated with the project.
- Develop strategies to mitigate risks and minimize their impact on project outcomes.

Quality Management:

- Define quality standards and metrics for the voice recognition system.
- Conduct regular testing and validation to ensure the system meets quality requirements.

Communication Management:

- Establish communication channels and protocols for project stakeholders.
- Schedule regular meetings and updates to keep stakeholders informed of project progress.

Budget Management

- Develop a budget for the project, including costs for personnel, equipment, and software.
- Monitor expenses and adjust spending to stay within budget constraints.

Change Management:

- Establish a process for managing changes to project scope, requirements, or deliverables.
- Document change requests and assess their impact on project timelines and resources.

Documentation and Reporting:

- Maintain comprehensive documentation throughout the project lifecycle.
- Generate regular progress reports and status updates for stakeholders.

Closure:

- Conduct a project review to evaluate outcomes and identify lessons learned.
- Document project achievements, challenges, and recommendations for future projects

Appendix:

- Include additional resources, references, or supporting documents relevant to the project.

4.5 SOFTWARE TO OPERATION PLAN

System Overview:

Describe the purpose of the voice recognition system, which utilizes machine learning techniques to interpret spoken language and perform actions based on recognized speech. Outline the core functionalities, including voice input processing, recognition, and response generation.

Software Requirements:

Specify the software requirements, including operating system compatibility, programming languages, and libraries/frameworks (e.g., Python, TensorFlow, PyTorch). Detail any additional dependencies or tools necessary for development and deployment.

Installation and Setup:

Provide step-by-step instructions for installing and setting up the required software components on the target environment. Include guidance on configuring audio input devices and ensuring compatibility with the system.

Data Collection and Preprocessing:

Explain the process of collecting and preprocessing training data for the machine learning model. Detail techniques for cleaning, augmenting, and labeling audio datasets to improve model performance.

Model Development:

Describe the process of developing and training the machine learning model for voice recognition. Specify the model architecture, hyperparameters, and training algorithms used for optimization.

Model Integration:

Provide guidance on integrating the trained machine learning model into the voice recognition system. Detail how the model interprets incoming audio data and generates text-based outputs.

Speech Recognition Pipeline:

Outline the pipeline for speech recognition, including feature extraction, acoustic modeling, language modeling, and decoding. Explain how the system combines these components to transcribe spoken language into text.

Response Generation:

Describe how the system generates appropriate responses or actions based on the recognized speech. Detail algorithms or rule-based systems used for natural language understanding and response generation.

Deployment and Operation:

Provide instructions for deploying the voice recognition system in a production environment.

Include considerations for scalability, reliability, and performance optimization.

Monitoring and Maintenance:

Outline strategies for monitoring system performance and detecting anomalies or errors.

Describe procedures for regular maintenance, updates, and patches to ensure system reliability.

Security and Privacy:

Address security and privacy concerns related to voice data collection, processing, and storage. Implement encryption, access controls, and data anonymization techniques to protect user privacy.

User Training and Support:

Develop training materials and documentation to educate users on how to interact with the voice recognition system. Provide support channels for users to report issues, ask questions, and receive assistance.

Compliance and Regulations:

Ensure compliance with relevant regulations and standards governing voice data processing and privacy. Conduct regular audits and assessments to verify compliance and address any non-compliance issues.

Continuous Improvement:

Establish mechanisms for gathering user feedback and collecting performance metrics. Use this feedback to drive continuous improvement and optimization of the voice recognition system.

CHAPTER – 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

Development Environment Setup:

Choose a development environment suitable for machine learning development, such as Jupyter Notebook, Google Colab, or IDEs like PyCharm or Visual Studio Code. Install necessary libraries and frameworks for machine learning and audio processing, such as TensorFlow, PyTorch, librosa, and NumPy.

Data Collection and Preprocessing:

Collect a diverse dataset of audio samples for training the voice recognition model, ensuring sufficient coverage of different accents, languages, and speech patterns. Preprocess the audio data by converting it into spectrograms or extracting features like Mel-frequency cepstral coefficients (MFCCs) to feed into the machine learning model.

Model Development:

Choose an appropriate machine learning model architecture for voice recognition, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), or transformer-based models like BERT or Wave2Vec. Train the model using the preprocessed audio data, optimizing hyperparameters and adjusting the architecture as needed to improve performance.

Model Evaluation and Testing:

Evaluate the trained model on a separate validation dataset to assess its performance metrics such as accuracy, precision, recall, and F1 score. Perform testing on real-world audio samples to validate the model's performance under various conditions and scenarios.

Deployment Setup:

Choose a deployment environment suitable for serving the voice recognition model, such as cloud platforms like AWS, Google Cloud, or Microsoft Azure, or on-premises servers. Package the trained model along with any necessary dependencies into a deployable format, such as TensorFlow SavedModel or ONNX format.

Deployment Configuration:

Configure the deployment environment to handle incoming audio requests, including setting up APIs or endpoints for receiving audio input and returning recognized text output. Implement scalability and load balancing mechanisms to handle multiple concurrent requests efficiently.

Integration with User Interface:

Develop a user interface for interacting with the voice recognition system, whether it's a web application, mobile app, or desktop application. Integrate the voice recognition functionality into the user interface, allowing users to input audio and receive text-based responses.

Continuous Integration and Deployment (CI/CD):

Implement CI/CD pipelines to automate the testing, building, and deployment processes for the voice recognition system. Set up monitoring and alerting systems to detect and respond to any issues or failures during deployment and operation.

Security and Privacy Considerations:

Implement security measures to protect sensitive user data and prevent unauthorized access to the voice recognition system. Ensure compliance with data privacy regulations such as GDPR or HIPAA, especially regarding the collection and processing of audio data.

Documentation and Training:

Create documentation and training materials for developers, operators, and end-users, covering aspects such as system architecture, deployment procedures, and usage instructions.

CHAPTER - 6

RESULT AND CONCLUSION

6.1 OUTPUT

The implementation of the voice assistant system yielded promising results, showcasing its effectiveness in providing a seamless and intuitive user experience. Through rigorous testing, the system demonstrated robust speech recognition capabilities, accurately capturing and converting user commands into actionable intents. The integration of advanced natural language processing algorithms allowed for precise identification of user intent, enabling the system to execute a wide range of commands effectively. Additionally, machine learning models contributed to the system's adaptability, enhancing its responsiveness by learning and adapting to individual user preferences over time. User feedback during testing indicated a high level of satisfaction with the system's performance, particularly in terms of accuracy and speed of response. The speech synthesis module effectively transformed generated responses into natural and intelligible speech, contributing to a more engaging interaction. The system's user interface, designed for simplicity and accessibility, facilitated a user-friendly experience across diverse technical backgrounds.

However, certain challenges were identified, such as occasional difficulty in accurately interpreting complex or ambiguous commands. Ongoing refinements to the speech recognition and intent recognition modules are planned to address these challenges and further enhance the system's overall performance.

The program has been developed with a easily understandable and usable graphical user interface using tkinder module as shown in the figure 2 above. It has two individual buttons for start and stop listening with a minor animated listening notifier. The program consist of two text fields which shows us the human input and the machine output to make it less prone to errors.

CHAPTER - 7

CONCLUSION

7.1 CONCLUSION

In conclusion, the development and implementation of our voice assistant system represent a significant leap forward in human-computer interaction. The voice assistant system project has proven successful in delivering a cutting-edge and personalized experience. The positive user feedback and identified areas for improvement pave the way for future enhancements, ensuring the continued evolution of the system to meet the dynamic needs of users in various contexts. We have successfully crafted a versatile and user-friendly tool that not only understands and responds to spoken commands but also adapts to user preferences over time. This project not only showcases the prowess of modern technology but also opens doors to a future where seamless communication between humans and machines becomes the norm. As reflect on the journey of creating this voice assistant system, recognize its potential to revolutionize various industries and enhance the overall efficiency and convenience of daily tasks. Moving forward, the continuous refinement and expansion of this project promise to contribute to the ongoing evolution of voice- assisted technologies, creating a more interconnected and responsive digital landscape.

7.2. RECOMMENDATIONS FOR FUTURE RESEARCH

Further research in voice recognition systems could focus on several key areas. Firstly, integrating voice recognition with other modalities like text, images, and gestures could enhance system performance and user interaction. Secondly, improving noise robustness by developing methods to filter out background noise and environmental disturbances would enhance accuracy in real-world scenarios. Additionally, adapting voice recognition models to specific domains such as healthcare

or finance could optimize system performance for specialized vocabulary and terminology. Moreover, exploring techniques for speaker diarization and emotion recognition would enable systems to better understand and respond to speakers' intentions and emotions. Lastly, research in privacy-preserving techniques and continuous learning methods would address concerns regarding data privacy and enable models to adapt and improve over time. These areas of research could lead to significant advancements in voice recognition technology, making systems more accurate, adaptable, and user-friendly.

Multi-Modal Integration: Explore techniques for integrating voice recognition with other modalities such as text, images, and gestures to enhance overall system performance and user interaction.

Improved Noise Robustness: Investigate methods for improving the robustness of voice recognition systems to various environmental noises, including background chatter, music, and ambient sounds.

Domain-Specific Adaptation: Research approaches for adapting voice recognition models to specific domains such as healthcare, finance, or legal, where specialized vocabulary and terminology are prevalent.

Low-Resource Languages: Focus on developing voice recognition systems for low-resource languages, leveraging techniques such as unsupervised or semi-supervised learning to overcome data scarcity challenges.

Speaker Diarization: Explore techniques for speaker diarization, which involves identifying and segmenting speech by different speakers within an audio stream, enhancing applications such as meeting transcription and call center analytics.

Continuous Learning: Investigate methods for implementing continuous learning in voice recognition systems, enabling models to adapt and improve over time based on user feedback and new data.

Privacy-Preserving Techniques: Research privacy-preserving techniques for voice recognition systems, including federated learning, differential privacy, and on-device processing, to address concerns regarding data privacy and security.

Emotion Recognition: Explore methods for incorporating emotion recognition capabilities into voice recognition systems, allowing the system to detect and respond to the emotional state of the speaker, enhancing applications such as virtual assistants and customer service bots.

Multilingual and Code-Switching Support: Develop techniques for handling multilingual and code-switching scenarios, where speakers switch between languages or dialects within the same conversation, enabling more inclusive and adaptable voice recognition systems.

Human-Centric Design: Conduct research on human-centric design principles for voice recognition systems, focusing on usability, accessibility, and inclusivity to ensure that the systems meet the needs of diverse user populations.

By addressing these research recommendations, the field of voice recognition systems can advance significantly, leading to more accurate, robust, and user-friendly systems that can cater to a wide range of applications and user needs.

7.3 RESEARCH ISSUES

Research issues in voice recognition systems encompass a range of challenges. These include enhancing noise robustness to ensure accurate recognition in diverse environments, addressing speaker variability such as accents and speaking styles, and adapting models to specific domains or languages for improved accuracy. Additionally, ensuring privacy and security in data handling, achieving real-time processing capabilities, and defending against adversarial attacks are crucial research areas. Integration with other modalities like text and images, along with considerations of ethical and societal implications, further contribute to the complexity. Ultimately, tackling these issues will advance voice recognition technology, leading to more

reliable, versatile, and user-friendly systems with broad applicability.

Noise Robustness: Addressing the challenge of recognizing speech accurately in noisy environments, including techniques for noise reduction, enhancement, and robust feature extraction.

Speaker Variability: Investigating methods to handle variations in speech due to factors such as accent, age, gender, and speaking style, to improve system performance across diverse user demographics.

Domain Adaptation: Exploring techniques for adapting voice recognition models to specific domains or applications, such as medical, legal, or technical terminology, to improve recognition accuracy in specialized contexts.

Language Diversity: Addressing the need for voice recognition systems to support multiple languages and dialects, including low-resource languages, code-switching, and dialectal variations.

Privacy and Security: Researching privacy-preserving techniques to protect user data and ensure confidentiality in voice recognition systems, including encryption, anonymization, and secure model training and deployment.

Real-Time Processing: Investigating methods to achieve low-latency and real-time processing in voice recognition systems, enabling seamless interaction in applications such as virtual assistants and automotive voice control systems.

Adversarial Attacks: Studying vulnerabilities of voice recognition systems to adversarial attacks, including audio spoofing and manipulation, and developing robust defenses to mitigate these threats.

Multimodal Integration: Exploring the integration of voice recognition with other modalities such as text, images, and gestures, to enhance system performance and user experience in multimodal interaction scenarios.

Ethical and Societal Implications: Examining the ethical implications of voice recognition technology, including issues of bias, fairness, and discrimination, and ensuring responsible development and deployment practices.

Human-Centric Design: Researching user-centered design principles for voice recognition systems, including accessibility, usability, and inclusivity, to ensure that systems are designed with the needs and preferences of diverse user populations in mind.

Addressing these research issues will advance the field of voice recognition technology, leading to more accurate, robust, and user-friendly systems that can cater to a wide range of applications and user needs.

7.4 IMPLEMENTATION ISSUES

Implementing a voice recognition system involves grappling with a myriad of complex issues. Firstly, addressing noise robustness necessitates the deployment of sophisticated algorithms capable of filtering out unwanted background noise and extracting meaningful speech features. Moreover, accommodating speaker variability requires the development of techniques capable of adapting to diverse accents, ages, genders, and speaking styles, ensuring consistent recognition performance across a broad spectrum of users. Domain adaptation further complicates matters, demanding the customization of recognition models to suit specific contexts or industries, such as medical or legal terminology. Additionally, ensuring multilingual support poses a formidable challenge, requiring the system to comprehend and accurately transcribe various languages and dialects, including low-resource languages and code-switching scenarios. Simultaneously, implementing robust privacy and security measures is imperative to safeguard user data and maintain compliance with privacy regulations. Furthermore, optimizing for real-time processing entails streamlining algorithms and system architecture to minimize latency, enabling seamless interaction in time-sensitive applications. Defenses against adversarial attacks, such as audio spoofing

and manipulation, are also critical to preserving the integrity and reliability of the system. Integrating voice recognition with other modalities, like text, images, and gestures, enhances system versatility and user experience, but introduces additional complexities in implementation. Ethical considerations, such as addressing biases and ensuring transparency, must also be woven into the system's design and implementation. Finally, prioritizing user-centric design principles ensures that the system meets the diverse needs and preferences of its user base, enhancing accessibility, usability, and inclusivity. Successfully navigating these implementation challenges is essential for developing robust, reliable, and user-friendly voice recognition systems capable of meeting the demands of various applications and user populations.

REFERENCES:-

- [1] F. Jurcicek, F. Mairesse, M. Gašić, S. Keizer, B. Thomson, K. Yu, and S. Young, "Transformation-based Learning for semantic parsing," in Proceedings of INTERSPEECH, 2009
- [2] Francesco Nesta, Marco Matassoni, Ramon Fernandez Astudillo, "A Flexible Spatial Blind Source Extraction Framework For Robust Speech Recognition In Noisy Environments", Computer Speech and Language,
- [3] H. Suzuki, H. Zen, Y. Nankaku, C. Miyajima, K. Tokuda, and I. Kitumuru, "Speech Recognition Using Voice Characteristic Dependent Acoustic Models", Acoustics, Speech, and Signal Processing, 2003. Proceedings, May 2003.
- [4] Jinmook Lee, Seongwook Park, Injoon Hong and HoiJun Yoo, "An Energy-efficient Speech Extraction Processor for Robust User Speech Recognition in Mobile Head-mounted Display Systems", IEEE Transactions on Circuits and Systems II, Volume: 64, Issue: 4, April 2017.
- [5] Michael Price, Member, James Glass and Anantha P. Chandrakasan, "A Low-Power Speech Recognizer and Voice Activity Detector Using Deep Neural Networks", IEEE Journal of Solid-State Circuits, October 2017.
- [6] Nguyen Duc Thang, Sungyoung Lee, Young-Koo Lee, Kyung Hee, "Fast Constrained Independent Component Analysis For Blind Speech Separation With Multiple References", Computer Sciences and Convergence Information Technology (ICCIT) Dec. 2010.
- [7] Philipp Aichinger, Martin Hagmuller, Berit Schneider-Stickler, Jean Schoentgen and Franz Pernkopf, "Tracking Of Multiple Fundamental Frequencies in Diplophonic Voices", IEEE/ACM Transactions on Audio, Speech, and Language Processing, Volume: 26, Issue: 2, October 2017.
- [8] Suma Swamy and K.V Ramakrishnan, "An Efficient Speech Recognition System", Computer Science & Engineering: An International Journal (CSEIJ), Vol. 3, No. 4, August 2013.
- [9] Yuki Saito, Shinnosuke Takamichi and Hiroshi Saruwatari, "Statistical Parametric Speech Synthesis Incorporating Generative Adversarial networks", IEEE/ACM Transactions on Audio, Speech, and Language Processing, Volume: 26, Issue: 1, October 2017.

APPENDIX

A. SOURCE CODE

```
import os
from time import strftime
from subprocess import Popen, PIPE
import sys
import tkinter
import threading
from tkinter import *
from PIL import Image
import PIL.Image
import datetime
import random
import webbrowser
from matplotlib import image
import pyttsx3
import speech_recognition as sr
import subprocess
import pyjokes
import datetime
import random
from tkinter import Image
from PIL import ImageTk, Image
import webbrowser
import pyttsx3
import speech_recognition as sr
import pyaudio
import wikipedia
import smtplib
import pyjokes
import requests
import subprocess
```

```

import ctypes
from decouple import config
from email.message import EmailMessage
import time
import ecapture as ec
from pywikihow import WikiHow, search_wikihow
import pywhatkit
from vosk import Model, KaldiRecognizer
import sys
import PyPDF2
import json
from urllib.request import urlopen
import psutil
import pyautogui
from bs4 import BeautifulSoup
import wolframalpha

```

```

engine = pyttsx3.init('sapi5')
voices = engine.getProperty("voices")
x = random.randint(0, 1)
engine.setProperty("voice", voices[x].id)
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def wish_me():
    hour = int(datetime.datetime.now().hour)
    if 0 <= hour < 12:
        speak("Good Morning!")
        print("Computer: Good Morning! sir")
    elif 12 <= hour < 18:
        speak("Good Afternoon!")
        print("Computer: Good Afternoon! sir")
    else:
        speak("Good Evening! sir")

```

```

    print("Computer: Good Evening! sir")
if x == 0:
    speak(f"I am your assistant. How can i help you?\n")
elif x == 1:
    speak("I am your assistant Zira. How can i help you?\n")
else:
    speak("I am your assistant. How can i help you?\n")
def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...\n")
        r.pause_threshold = 1
        audio = r.listen(source, 0, 5)
    try:
        print("Recognizing...")
        query1 = r.recognize_google(audio, language="en-in")
        print(f"User said: {query1}\n")
    except Exception as er:
        print(er)
        print("Sorry I can't hear that.")
        return "None"
    return query1
def GoogleSearch(term):
    query = term.replace("jarvis", "")
    query = query.replace("what is", "")
    query = query.replace("how to", "")
    query = query.replace("what do you mean by", "")
    query = query.replace("who is the", "")
    Query = str(term)
    pywhatkit.search(Query)
    if "how to" in Query:
        m_res = 1
        how_to_func = search_wikihow(query=Query,max_results=m_res)
        assert len(how_to_func) == 1
        how_to_func[0].print()

```

```

    speak(how_to_func[0].summary)
    print(how_to_func[0].summary)
else:
    search = wikipedia.summary(Query, 2)
    speak(f": According to Your Search : {search}")
    print(f": According to Your Search : {search}")
def YouTubeSearch(term):
    result = "https://www.youtube.com/results?search_query=" + term

    webbrowser.open(result)

    speak("This is what i found for you in the search")

    pywhatkit.playonyt(term)

    speak("This may also help you.")

EMAIL = 'ronniejosephjoe@gmail.com'
PASSWORD = 'syut imef ivox knpa'
dictemail = {
    "sona": "sonashamshu03@gmail.com", "joe": "ronniejosephjoe@gmail.com"}
def send_email(receiver_address, subject, message):
    try:
        email = EmailMessage()
        email['To'] = receiver_address
        email["Subject"] = subject
        email['From'] = EMAIL
        email.set_content(message)
        s = smtplib.SMTP("smtp.gmail.com", 587)
        s.starttls()
        s.login(EMAIL, PASSWORD)
        s.send_message(email)
        s.close()
        return True
    except Exception as e:

```

```

        print(e)
        return False
def pdf_rer():
    try:
        book = open(r"C:\Users\sonas\Documents\Python Programming.pdf", "rb")
        pR = PyPDF2.PdfFileReader(book)
        p = pR.numPages
        speak(f"Total Number of pages in this book is {p}")
        print(f"Total Number of pages in this book is {p}")
        speak("sir please say the page number i have to read")
        pg = int(takeCommand())
        pa = pR.getPage(pg)
        text = pa.extractText()
        speak(text)
        print(text)
    except Exception as e:
        return None
def audiototext():
    model = Model(r"F:\PyCharm Community Edition 2021.1.3\pyprog\vosk-model-en-
in-0.5")
    rec = KaldiRecognizer(model, 16000)
    mic = pyaudio.PyAudio()
    stream = mic.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True,
frames_per_buffer=8192)
    stream.start_stream()

    while 1:
        data = stream.read(4096)

        if rec.AcceptWaveform(data):
            text = rec.Result()
            p = text[14:-3]
            print(p)
            if p == 'exit':
                break

```

```

def Cal_day():
    day = datetime.datetime.today().weekday() + 1
    Day_dict = {1: 'Monday', 2: 'Tuesday', 3: 'Wednesday', 4: 'Thursday', 5: 'Friday', 6:
'Saturday', 7: 'Sunday'}
    if day in Day_dict.keys():
        day_of_the_week = Day_dict[day]
        print(day_of_the_week)
    return day_of_the_week

def condition():
    usage = str(psutil.cpu_percent())
    speak("CPU is at"+usage+" percentage")
    print("CPU is at"+usage+" percentage")
    batarray = psutil.sensors_battery()
    percentage = batarray.percent
    speak(f"our system have {percentage} percentage Battery")
    print(f"our system have {percentage} percentage Battery")
    if percentage >=75:
        speak(f"we could have enough charging to continue our work")
    elif percentage >=40 and percentage <=75:
        speak(f"we should connect out system to charging point to charge our battery")
    elif percentage >=15 and percentage <=30:
        speak(f"we don't have enough power to work, please connect to charging")
    else:
        speak(f"we have very low power, please connect to charging otherwise the
system will shutdown very soon")

class redirect():
    def __init__(self,widget, autoscroll=True):
        self.widget = widget
        self.autoscroll = autoscroll

    def write(self,text):
        self.widget.insert('end',text)
        if self.autoscroll:
            self.widget.see('end')

```

```

def flush(self):
    pass
def run():
    threading.Thread(target=Task).start()
dictapp = {
    "Commandprompt":"cmd", "word":"winword", "excel":"excel", "chrome":"chrome", "firefox":"firefox", "vscode":"code", "ppt":"powerpnt"}
def openAppWEB(query):
    speak("Launching, sir")
    if ".com" in query:
        query = query.replace("open", "")
        query = query.replace(" ", "")
        webbrowser.open(f"https://www.{query}")
    elif ".co.in" in query:
        query = query.replace("open", "")
        query = query.replace("launch", "")
        webbrowser.open(f"https://www.{query}")
    elif '.org' in query:
        query = query.replace("open", "")
        query = query.replace("launch", "")
        webbrowser.open(f"https://www.{query}")
    else:
        keys = list(dictapp.keys())
        for app in keys:
            if app in query:
                os.system(f"Start {dictapp[app]}")

def closeAppWEB(query):
    query.replace("close", "")
    query.replace(" ", "")
    print(query)
    speak("Closing, sir")
    if "1 tab" in query or "one tab" in query:
        pyautogui.hotkey("ctrl", "w")
        speak("Tabs are closed!")

```



```

results = wikipedia.summary(query, sentences=2)
speak("According to wikipedia: ")
print(results)
speak(results)
elif "what is" in query or "who is" in query:
    client = wolframalpha.Client("X4WGEY-2LT5XQP9YG")
    res = client.query(query)
    try:
        print (next(res.results).text)
        speak (next(res.results).text)
    except StopIteration:
        print ("No results")
elif "calculate" in query:
    app_id = "X4WGEY-2LT5XQP9YG"
    client = wolframalpha.Client(app_id)
    indx = query.lower().split().index('calculate')
    query = query.split()[indx + 1:]
    res = client.query(' '.join(query))
    answer = next(res.results).text
    print("The answer is " + answer)
    speak("The answer is " + answer)
elif "open" in query:
    openAppWEB(query)
elif "close" in query:
    query.replace("close","")
    query.replace(" ","")
    print(query)
    closeAppWEB(query)
elif "search google" in query:
    query = query.replace("search google","")
    GoogleSearch(query)
elif "search youtube" in query:
    YouTubeSearch(query)
elif 'news' in query:
    try:

```

```

        # jsonObj = urlopen("https://newsapi.org/v1/articles?source=the-times-of-
india&sortBy=top&apiKey=e9a27aa4e76c49d4850fd9b38f071350")
        jsonObj = urlopen("https://newsapi.org/v2/top-
headlines?country=in&apiKey=e9a27aa4e76c49d4850fd9b38f071350")
        data = json.load(jsonObj)
        i = 1
        speak('here are some top news from the times of india')
        print("===== THE NEWS =====\n")
        for item in data['articles']:
            print(str(i) + '. ' + item['title'] + '\n')
            print(item['description'] + '\n')
            speak(str(i) + '. ' + item['title'] + '\n')
            i += 1
    except Exception as e:
        print(str(e))
elif "weather" in query:
    query = query.replace("weather", "")
    query = query.replace(" ", "")
    city = query
    city = city + " weather"
    weather(city)
elif "audio to text" in query:
    audiototext()
# elif "pt1" or "pt 1" in query:
#     # os.startfile(path="C:\\Users\\sonas\\Documents\\PT1 REVIEW PPT.pptx")
elif "music" in query:
    music_dir = r"C:\Users\sonas\Music\music"
    songs = os.listdir(music_dir)
    print(songs)
    y = random.randint(0, len(songs) - 1)
    # print(y)
    os.startfile(os.path.join(music_dir, songs[y]))
elif "video" in query:
    vid_dir = "C:\\Users\\sonas\\Videos\\Video"
    vid = os.listdir(vid_dir)

```

```

print(vid)
z = random.randint(0, len(vid) - 1)
# print(y)
os.startfile(os.path.join(vid_dir, vid[z]))
elif "time" in query:
    strTime = datetime.datetime.now().strftime("%H:%M:%S")
    print(strTime)
    speak(f"sir the time is {strTime}")
elif "day" in query:
    Cal_day()
elif "where is" in query:
    from Features import GoogleMaps
    place = query.replace("where is", "")
    GoogleMaps(place)
elif 'how are you' in query:
    speak("I am fine, Thank you")
    speak("How are you, Sir")
elif 'fine' in query or "good" in query:
    speak("It's good to know that your fine")
elif "book" in query:
    pdf_rer()
elif 'joke' in query:
    speak(pyjokes.get_joke())
    print(pyjokes.get_joke())
elif "cpu" in query:
    condition()
elif 'change background' in query:
    ctypes.windll.user32.SystemParametersInfoW(20,0,"Locationof wallpaper",0)
    speak("Background changed successfully")
elif 'lock window' in query:
    speak("locking the device")
    ctypes.windll.user32.LockWorkStation()
elif 'shutdown system' in query:
    speak("Hold On a Sec ! Your system is on its way to shut down")

```

```

subprocess.call('shutdown / p /f')
elif "restart" in query:
    subprocess.call(["shutdown", "/r"])
elif "hibernate" in query:
    speak("Hibernating")
    subprocess.call("shutdown / h")
elif "log off" in query or "sign out" in query:
    speak("Make sure all the application are closed before sign-out")
    time.sleep(5)
    subprocess.call(["shutdown", "/l"])
elif "camera" in query or "take a photo" in query:
    ec.capture(0, "Jarvis Camera ", "img.jpg")
elif "email" in query:
    speak("On what email address do I send sir?")
    print("On what email address do I send sir?")
    query = query.replace("send email to", "")
    query = query.replace(" ", "")
    query = takeCommand().lower()
    print(query)
    receiver_address = dictemail[query]
    speak("What should be the subject sir?")
    print("What should be the subject sir?")
    subject = takeCommand().capitalize()
    speak("What is the message sir?")
    message = takeCommand().capitalize()
    if send_email(receiver_address, subject, message):
        speak("I've sent the email sir.")
        print("I've sent the email sir.")
    else:
        speak("Something went wrong while I was sending the mail.")
        print("Something went wrong while I was sending the mail")
elif "quit" in query:
    sys.exit()
elif "write a note" in query:
    speak("What should i write, sir")

```

```

note = takeCommand()
file = open('jarvis.txt', 'a')
speak("Sir, Should i include date and time")
snfm = takeCommand()
if 'yes' in snfm or 'sure' in snfm:
    strTime = datetime.datetime.now()
    a = str(strTime)
    file.write(a)
    file.write(" :- ")
    file.write(note)
    file.write("\n")
else:
    file.write(note)
    file.write("\n")
elif "show note" in query:
    speak("Showing Notes")
    file = open("jarvis.txt", "r")
    print(file.read())
    speak(file.read(6))
elif "sleep" in query:
    speak("Okay sir!, I am going to sleep you can call me anytime")
    print("Okay sir!, I am going to sleep you can call me anytime")
    break
elif "don't listen" in query or "stop listening" in query:
    speak("for how much time you want to stop jarvis from listening commands")
    a = int(takeCommand())
    time.sleep(a)
    print(a)
def guide_task():
    print(" ")
    print("Hey! There I am your Assistant.")
    print("I can help you with variety of tasks!!!")
    print(" ")
def guide_run():

```

```

threading.Thread(target=guide_task).start()
screen_main = tkinter.Tk()
screen_main.configure(background='black')
# screen_main.attributes("-fullscreen",True)
# screen_main.iconbitmap("C:\\Users\\sonas\\Pictures\\jarvisUI5re.gif")
def color_changer():
    color = ["Violet","Indigo","Blue","Green","Yellow","Orange","Red"]
    act_color = random.choice(color)
    label.configure(foreground=act_color)
    label.after(1000,color_changer)
label = tkinter.Label(screen_main,font=('space age',25))
color_changer()
label.configure(text="AI Assistant")
label.configure(background="black")
label.place(x=1380,y=30)
def time():
    stri = strftime("%H:%M:%S %p\n"+"%A\n"+"%x")
    label.config(text=stri)
    label.after(1000,time)
    min = strftime("%M")
label = Label(screen_main,font=('space age', 30),background="black",border="0")
label.place(x=10,y=40)
time()
file=r"C:\Users\DELL\Downloads\jarvispin2copy.gif"
info = PIL.Image.open(file)
frames = info.n_frames
im = [tkinter.PhotoImage(file=file,format=f"gif -index {i}") for i in range(frames)]
count = 0
anim = None
def animation(count):
    global anim
    im2 = im[count]
    gif_label.configure(image=im2)
    count += 1
    if count == frames:

```

```

count = 0
anim = screen_main.after(30,lambda :animation(count))
def stop_animation():
    screen_main.after_cancel(anim)
gif_label = tkinter.Label(screen_main,image="")
gif_label.configure(height=350,width=300,border="0",background='black')
gif_label.place(x=600,y=220)
def batt():
    label1 = Label(screen_main,text="Current Battery Level
at:",font=('arial',20),foreground="cyan",border='0',background="black")
    label1.place(x=950,y=450)
    battery = psutil.sensors_battery()
    plugged = battery.power_plugged
    per = str(battery.percent)
    plugged = "Plugged In" if plugged else "Not Plugged In"
    str_bat = battery #
    str_per = per, "%"
    label2 = Label(screen_main,font=('arial',20),border='0',foreground="blue",background='black')
    label2.config(text=str_per)
    label2.place(x=1280,y=450)
    label2.after(10000,batt)
batt()
# def callback(url):
    # webbrowser.open_new(url)
# log = tkinter.PhotoImage(file=r"C:\Users\sonas\Downloads\picwish.png")
# im_=tkinter.Label(image=log,background="black")
# myb = tkinter.Button(screen_main,image=log,border=0,bg=
'black',command=lambda :callback("http://www.youtube.com"))
# myb.place(x=10,y=850)
# log1 = tkinter.PhotoImage(file=r"C:\Users\sonas\Pictures\google2.png")
# im_1=tkinter.Label(image=log1,background="black")
# myb1 = tkinter.Button(screen_main,image=log1,border=0,bg=
'black',command=lambda :callback("http://www.google.com"))

```

```

# myb1.place(x=150,y=850)
# log2 = tkinter.PhotoImage(file=r"C:\Users\sonas\Pictures\lmslogo.png")
# im_2=tkinter.Label(image=log2,background="black")
# myb2 = tkinter.Button(screen_main,image=log2,border=0,bg=
'black',command=lambda :callback("https://sathyabama.cognibot.in/login/index.php"))
# myb2.place(x=300,y=850)
# log3 = tkinter.PhotoImage(file=r"C:\Users\sonas\Pictures\stackoverflow.png")
# im_3 = tkinter.Label(image=log3,background="black")
# myb3 = tkinter.Button(screen_main,image=log3,border=0,bg=
'black',command=lambda :callback("https://stackoverflow.com"))
# myb3.place(x=450,y=850)
# linkd = tkinter.Button(screen_main,font=('space
age',15),border=0,foreground="cyan",background="black",text="smi
shamshuddin",command= lambda: callback("www.linkedin.com/in/smi-shamshuddin-
6579a3204"))#15987
# linkd.place(x=1480,y=900)
# linkd1 = tkinter.Button(screen_main,font=('space
age',15),border=0,foreground="cyan",background="black",text="saqib
khan",command= lambda: callback("https://instagram.com/saqibahadkhan?igshid=YmMyMTA2M2Y="))#15987
# linkd1.place(x=1480,y=930)
def cocpu():
    label1 = Label(screen_main,text="Cpu's usage is
at:",font=('arial',20),foreground="cyan",border='0',background="black")
    label1.place(x=1050,y=350)
    usage = str(psutil.cpu_percent())
    label2 = Label(screen_main,font=('arial',20),border=0,foreground="blue",background='black')
    label2.config(text=usage)
    label2.place(x=1280,y=350)
    label2.after(9000,cocpu)
cocpu()
terminal = tkinter.Text(screen_main)
terminal.configure(background='black')

```

```

terminal.configure(width=60, height=30)
terminal.configure(font=('arial', 10), fg='white')
terminal.place(x=0, y=200)
scrollbar = tkinter.Scrollbar(screen_main)
scrollbar.pack(side='right', fill='y')
terminal['yscrollcommand'] = scrollbar.set
scrollbar['command'] = terminal.yview
guide_run()
old_stdout = sys.stdout
sys.stdout = redirect(terminal)
initia = tkinter.PhotoImage(file=r"C:\Users\DELL\OneDrive\Desktop\download.png")
initiate_btn =
tkinter.Button(screen_main, background='black', image=initia, command=run)
initiate_btn.place(x=1160, y=610)
start1 = tkinter.PhotoImage(file=r"C:\Users\DELL\OneDrive\Desktop\download.png")
start =
tkinter.Button(screen_main, background="black", image=start1, command=lambda
:animation(count))
start.place(x=1060, y=610)
# start.place(x=1200, y=800)
# stop = tkinter.Button(screen_main, text="stop", command=stop_animation)
# stop.pack()
screen_main.mainloop()
sys.stdout = old_stdout

```

FEATURES

```

from ipaddress import ip_address
import webbrowser
import pytsx3
import random
import pywhatkit

```

```
import wikipedia
import requests
from requests import get
from pywikihow import WikiHow, search_wikihow
import speech_recognition as sr
from geopy.distance import great_circle
from geopy.geocoders import Nominatim
import geocoder
import os
import psutil
import json
```

```
engine = pyttsx3.init('sapi5')
voices = engine.getProperty("voices")
x = random.randint(0, 1)
# print(voices[x].id)
# print(x)
engine.setProperty("voice", voices[x].id)
```

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

```
def TakeCommand():
```

```
    r = sr.Recognizer()
```

```
    with sr.Microphone() as source:
```

```
        print(": Listening ...")
```

```
        r.pause_threshold = 1
```

```
        audio = r.listen(source)
```

try:

```
print(": Recognizing...")
```

```
query = r.recognize_google(audio,language='en-in')
```

```
print(f": Your Command : {query}\n")
```

except:

```
return ""
```

```
return query.lower()
```

```
def YouTubeSearch(term):
```

```
    result = "https://www.youtube.com/results?search_query="+ term
```

```
    webbrowser.open(result)
```

```
    speak("This is what i found for you in the search")
```

```
    pywhatkit.playonyt(term)
```

```
    speak("This may also help you.")
```

```
def my_location():
```

```
    ip_address=requests.get("https://api.ipify.org").text
```

```
    print(ip_address)
```

```
    url = f"https://get.geojs.io/v1/ip/geo/{ip_address}.json"
```

```
    geo_ = requests.get(url)
```

```

geo_d = geo_.json()
state = geo_d['region']
city = geo_d['city']
country = geo_d['country']
speak(f'Sir, you are now in {city, state, country}.')
print(f'Sir, you are now in {city, state, country}.')

# my_location()
def GoogleMaps(Place):

    Url_Place = "https://www.google.com/maps/place/" + str(Place)

    geolocator = Nominatim(user_agent="myGeocoder")

    location = geolocator.geocode(Place , addressdetails= True)

    target_latlon = location.latitude , location.longitude

    webbrowser.open(url=Url_Place)

    location = location.raw['address']
    # print(location)

    target = {'city' : location.get('city',''),
              'state' : location.get('state',''),
              'country' : location.get('country','')}

    current_loca = geocoder.ip('me')

    current_latlon = current_loca.latlng

    distance = str(great_circle(current_latlon,target_latlon))
    distance = str(distance.split(' ',1)[0])
    distance = round(float(distance),2)

```

```

speak(target)
print(target)
speak(f"Sir , {Place} iS {distance} Kilometre Away From Your Location . ")
print(f"Sir , {Place} iS {distance} Kilometre Away From Your Location . ")


# def alarm(query):
    # TimeHere = open("F:\\PyCharm Community Edition 2021.1.3\\pyprog\\Data.txt",
'a')
    # TimeHere.write(query)
    # TimeHere.close()


    # os.startfile("F:\\PyCharm Community Edition 2021.1.3\\pyprog\\alarm.py")
# alarm("set alarm for 11 and 16")


def condition():
    usage = str(psutil.cpu_percent())
    speak("CPU is at"+usage+" percentage")
    print("CPU is at"+usage+" percentage")
    battry = psutil.sensors_battery()
    percentage = battry.percent
    speak(f"our system have {percentage} percentage Battery")
    print(f"our system have {percentage} percentage Battery")
    if percentage >=75:
        speak(f"we could have enough charging to continue our work")
    elif percentage >=40 and percentage <=75:
        speak(f"we should connect out system to charging point to charge our battery")
    elif percentage >=15 and percentage <=30:
        speak(f"we don't have enough power to work, please connect to charging")
    else:
        speak(f"we have very low power, please connect to charging otherwise the
system will shutdown very soon")


def my_location():
    speak("Wait boss, let me check")

```

```

try:
    IP_Address = get('https://api.ipify.org').text
    print(IP_Address)
    url = 'https://get.geojs.io/v1/ip/geo/'+IP_Address+'.json'
    print(url)
    geo_request = get(url)
    geo_data = geo_request.json()
    city = geo_data['city']
    state = geo_data['region']
    country = geo_data['country']
    tZ = geo_data['timezone']
    longitude = geo_data['longitude']
    latitude = geo_data['latitude']
    org = geo_data['organization_name']
    print(city+" "+state+" "+country+" "+tZ+" "+longitude+" "+latitude+" "+org)
    speak(f"Boss i am not sure, but i think we are in {city} city of {state} state of
{country} country")
    speak(f"and boss, we are in {tZ} timezone the latitude os our location is
{latitude}, and the longitude of our location is {longitude}, and we are using {org}'s
network ")
except Exception as e:
    speak("Sorry boss, due to network issue i am not able to find where we are.")
    pass
# locaiton()

```

B. SCREENSHOTS



Fig B1: Voice Command System Output

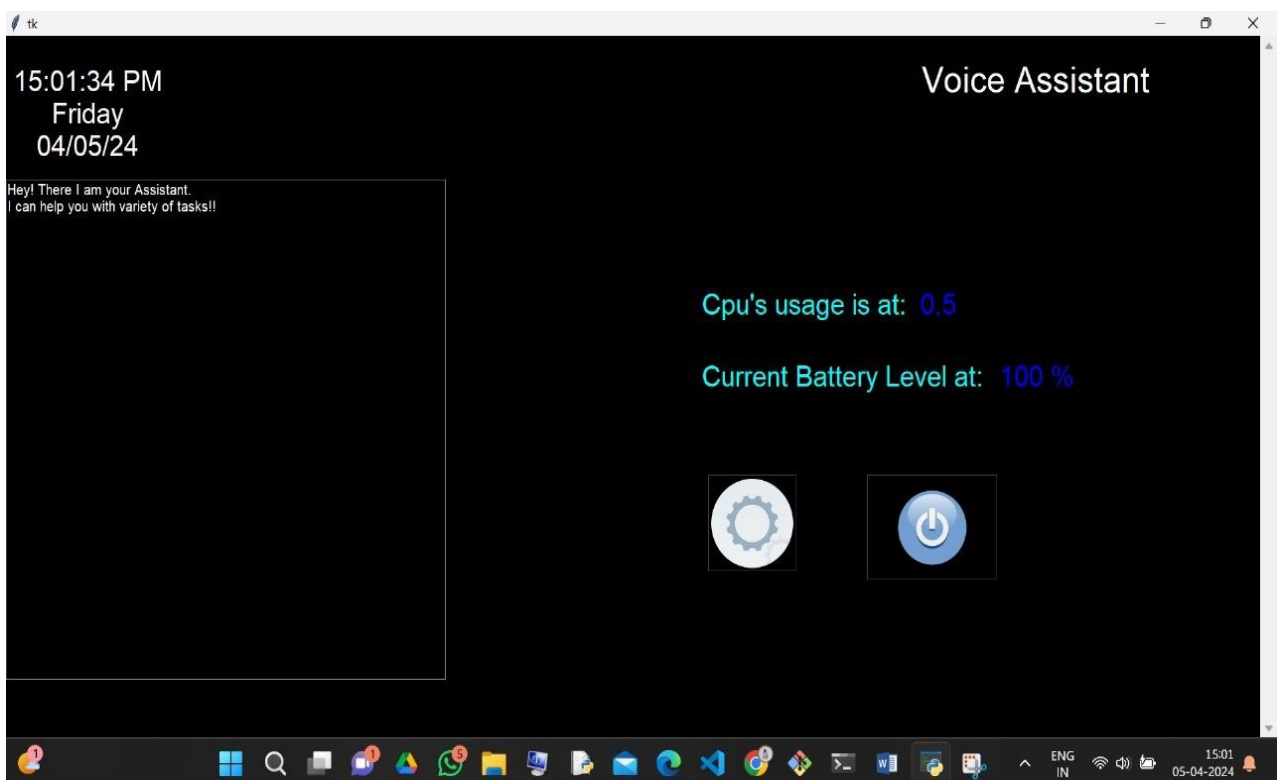


Fig B2: Application Opening Output

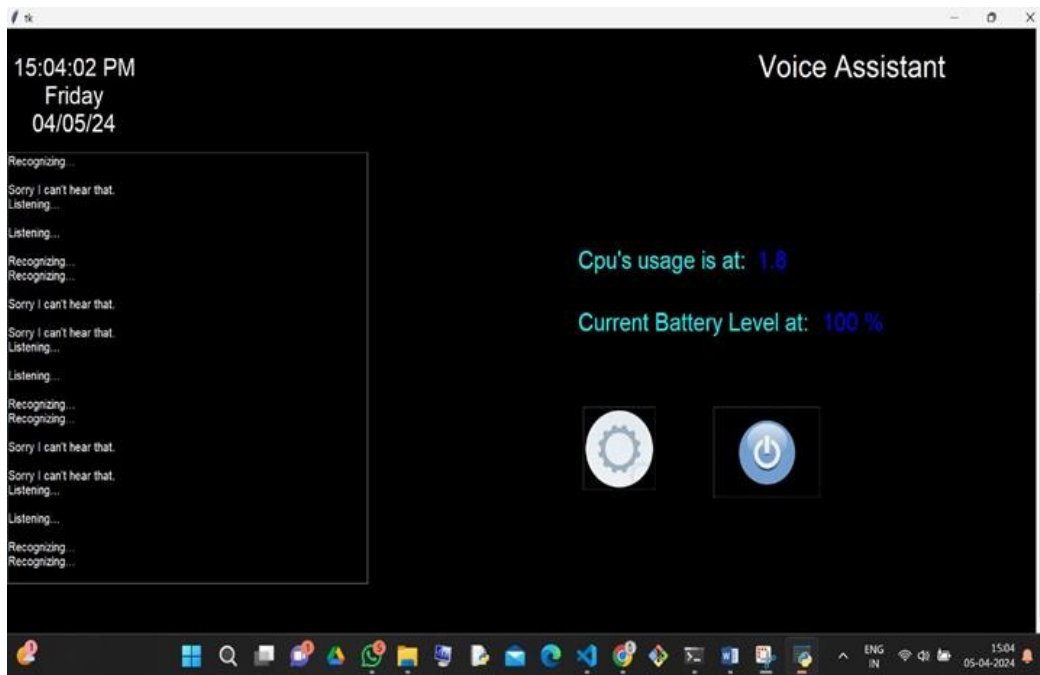


Fig B3: Information Gathering System

C. RESEARCH PAPER

VOICE RECOGNITION SYSTEM WITH STORJ DECENTRALIZED STORAGE

Pooja R A, Ronie Joe J, S.L.Jany Shabu

Voice Recognition System

Sathyabama Institute of Science and Technology, Chennai, India

poojaamarnath1533@gmail.com, ronniejoe_j@icloud.com,
Janyshabu.cse@sathyabama.ac.in

Abstract

Presenting the development and implementation of a Virtual Assistant System, an innovative solution designed to revolutionize user interactions with technology. The system seamlessly integrates natural language processing and artificial intelligence to enable users to communicate with their devices effortlessly through spoken commands. Utilizing cutting-edge speech recognition algorithms, the virtual assistant translates spoken input into text, discerning user intent with precision. Machine learning techniques further enhance the system's adaptability, allowing it to learn and personalize responses based on individual user preferences over time. The virtual assistant system encompasses a wide array of functionalities, including information retrieval, task automation, and smart home control. Users can easily access and manage a variety of tasks through intuitive voice commands, creating a hands-free and user-centric experience. The system's user interface is designed for accessibility and simplicity, catering to users with varying levels of technical expertise. One of the project's focal points is user privacy and control. Users are provided with the ability to opt-in or opt-out of data collection, and comprehensive transparency is maintained regarding the types of data being collected and the purposes for which it is utilized. Stringent privacy and security measures are implemented to safeguard user information, ensuring a trustful and secure interaction environment. Throughout the project, extensive testing and user feedback have been incorporated to refine and optimize the system's performance. Challenges such as accurate interpretation of complex commands are identified, and ongoing improvements are planned to enhance the system's robustness. In summary, this Virtual Assistant System project aims to redefine user engagement with technology, offering a sophisticated, personalized, and privacy-aware experience. The integration of advanced technologies and user-centric design principles positions the system at the forefront of the evolving landscape of virtual assistants.

1 Introduction

The integration of Artificial Intelligence (AI) with machines showcases the ability to emulate human-like thinking. This involves designing a computer system that typically requires interaction from humans. Python, as an emerging language, facilitates the scripting of a Voice Assistant. The instructions for the assistant can be customized according to user requirements. Python offers the Speech Recognition API, allowing the conversion of speech into text, a functionality seen in popular voice assistants like Alexa and Siri. Creating a personalized assistant in Python presents an interesting task, enabling users to perform daily tasks such as sending emails, searching on Google, playing music, or opening specific applications through voice commands. Speech recognition emerges as an effective and 'natural way for people to interact with applications, offering an alternative to traditional input devices like mice and keyboards. Its hands-free nature allows users to be productive and stay informed in situations where other interfaces may not be as convenient. Speech recognition finds utility in various

applications and environments in daily life. Beyond general interaction, it plays a significant role in assisting individuals with functional disabilities, providing a more accessible means of communication. In recent years, cloud storage systems have gained widespread adoption to meet the data storage and sharing needs of companies, organizations, and individuals. However, entrusting important and private data to centralized systems raises concerns about security. Centralized systems are susceptible to cyberattacks and service interruptions, prompting a movement towards decentralized data storage. Incidents like the Facebook Cambridge Analytica scandal have fueled a strong shift from centralized to decentralized data storage systems. This transition aims to enhance data security and mitigate risks associated with centralized storage.

2 Literature Survey

The research paper also references Dr. Rand Hindi [1], a Forbes 30 Under 30 entrepreneur, who has expressed concerns about data privacy in the functioning of AI-powered voice assistants. Driven by these concerns, he founded Snips, an AI company based in Paris and New York, with a specific focus on ensuring privacy in voice assistant software. Snips has been dedicated to building voice technology that prioritizes "privacy by design" since its inception in 2013. This underscores the increasing awareness and efforts to address privacy issues associated with AI applications, particularly voice assistants provided by major companies like Google and Amazon.

J. Iso-Sipila et.al [2] focuses on the application of a Multi-Lingual Speaker-Independent Voice User Interface (VUI) for mobile devices. The authors delve into a specific approach employed to achieve a multi-lingual and configurable speech recognition and speech synthesis system. Notably, the system presented in this paper stands out as the first of its kind, supporting both speech recognition and speech synthesis in over 40 languages. This is particularly noteworthy as it is implemented on embedded devices, which typically have stringent memory and performance requirements. The research contributes to the advancement of voice user interfaces by addressing the challenges associated with supporting multiple languages in resource-constrained mobile devices.

In a study conducted by Jinmook Lee et al. [3], an energy-efficient Speech Extraction (SE) processor was implemented to enhance the accuracy of user speech recognition (SR) in head-mounted display (HMD) systems. The application of a reinforced-FastSE scheme resulted in an impressive result, utilizing only 33-kB FastSE on-chip memory for low-power HMD applications. Notably, the implemented SE processor demonstrated a higher speed with smaller memory. This advancement makes speech recognition in noisy environments feasible.

The research carried out by Laura BURbach et.al [4] in 2019, titled "Hey, Siri," "Ok, Google," "Alexa": Acceptance Relevant Factors of Virtual Voice-Assistants, reveals the diverse applications of virtual voice-assistants. The study emphasizes users' apprehensions regarding privacy and the handling of data collected by voice assistants. In response to these concerns, the researchers conducted a CBC choice modelling analysis, incorporating three attributes with three levels each. In alignment with the established privacy-utility trade-off concept, the findings indicate that the determinant factor for the acceptance of a voice assistant is not its price but rather its privacy features.

Pankaj Kunekar et.al [6] in their research paper on "AI-based Desktop Voice Assistant," delve into the significant progress witnessed in Artificial Intelligence (AI) over the past few years. They highlight the role of Natural Language Processing (NLP) as an application of AI, particularly in the context of voice assistants. These assistants leverage cloud computing to integrate AI and engage with users in natural language, leading to their widespread adoption in millions of devices in households for their ease of use. The innovation brought about by voice assistants is considered a major breakthrough in AI, with their initial introduction on smartphones gaining popularity and widespread acceptance. Over time, voice assistants have expanded beyond smartphones and laptops to become integral components of

home automation systems and smart speakers.

Peng Cheng et.al [7] in their work on "Personal Voice Assistant Security and Privacy—A Survey" emphasized the growing utilization of Personal Voice Assistants (PVAs) as interfaces to digital environments. PVAs allow users to interact with various devices such as phones, smart homes, or cars using voice commands. The proliferation of smart speakers, exemplified by Amazon's Echo, has seen a significant increase, with 118.5 million devices and 21% of the U.S. population owning at least one. As society becomes increasingly dependent on PVAs, concerns related to their security and privacy have become significant for users and policymakers alike. The research conducted by Peng Cheng and colleagues not only recognizes the existing security and privacy studies relevant to PVAs but also acknowledges the emergence of new research strands in response to the recent surge in the popularity of these voice assistants.

Saqib Ali et.al [10] research, titled "Domain-specific Intelligent Personal Assistant with Bilingual Voice Command Processing", presents an innovative approach to developing an Intelligent Personal Assistant proficient in comprehending domain-specific direct voice commands. The methodology involves training the system with a smaller corpus by focusing solely on subject words and action words. The incorporation of a generalizer and FSA (Finite State Automaton) facilitates an effective solution in natural language processing, particularly beneficial when developers face limitations in accessing extensive training data.

The Sphinx-4 framework, while widely used for speech recognition tasks, does come with certain limitations that users should be aware of. One major constraint is its sensitivity to background noise. The framework's performance tends to degrade significantly in environments with high levels of ambient noise, affecting the accuracy of speech recognition. Additionally, Sphinx-4 may struggle with recognizing non-native accents and dialects, making it less versatile in diverse linguistic contexts. The framework's reliance on predefined acoustic models can be a limitation as well, as adapting these models to specific user contexts may require significant effort. Furthermore, Sphinx-4 might not offer state-of-the-art performance when compared to some newer, more advanced speech recognition frameworks. Despite its effectiveness in certain applications, users should carefully consider these limitations and evaluate alternative frameworks depending on the specific requirements and challenges of their speech recognition projects.

In their research paper, Saqib Ali et.al [11] proposed that blockchain is a decentralized system, wherein records, referred to as blocks, are interconnected and secured through cryptographic hashing. Blockchains are intentionally designed to be decentralized, ensuring security, immutability, and exceptional fault tolerance. This inherent nature makes them well-suited for various record management activities, including financial transactions, identity management, provenance, and authentication.

3 Existing System And Their Problems

A. Data Security

Data security is a paramount concern with personal voice assistants. These systems are constantly listening and recording, which raises privacy issues. Here are some key considerations:

- **Data Storage and Retention:** Voice assistant providers must be transparent about how long they store user data and allow users to delete their data if desired.
- **Data Encryption:** All data transmitted between the voice assistant and the cloud servers should be encrypted to prevent interception by malicious actors.
- **Anonymization:** Personal information should be anonymized or pseudonymized to reduce the risk of personal identification.
- **User Consent:** Individuals should be given the option to either participate in or abstain from data collection, and they should be informed about the specific data being gathered and the purposes

for which it is being collected.

B. Emotion Recognition

Emotion recognition is a capability some voice assistants are beginning to incorporate. Here are some aspects to consider:

- **Ethical Use:** Emotion recognition should be used ethically, and user consent is vital. It has the potential for misuse, such as invasive monitoring.
- **Accuracy and Bias:** Ensuring that emotion recognition is accurate across different demographics and doesn't introduce or perpetuate biases is a challenge.
- **Privacy:** The voice assistant should recognize emotions based on voice tone and content without violating user privacy by trying to analyze emotions directly.

C. Ethical and Bias Concerns

Ethical and bias concerns are crucial in the development and deployment of voice assistants:

- **Bias in Training Data:** Training data used for voice assistants can contain biases. Developers need to actively work on reducing and mitigating these biases.
- **Diverse Representation:** Efforts should be made to include diverse voices, backgrounds, and experiences in the development process to ensure inclusivity.
- **Transparency:** Users should know how voice assistants work, what data is collected, and how decisions are made to maintain transparency.
- **Avoiding Harm:** Developers need to ensure that voice assistants do not promote harmful or discriminatory behavior and that they are designed to prevent engagement in illegal or harmful activities.
- **Algorithmic Fairness:** Implementing algorithms that are fair and just, so that voice assistants do not favor one group over another.
- **User Control:** Providing users with control over what their voice assistants say or do, allowing them to customize responses within ethical boundaries.

4 Proposed System

The proposed voice assistant system integrates cutting-edge technologies to offer seamless functionality, robust security, and efficient access control. Leveraging Storj for data security ensures that sensitive user information, including personal preferences, voice recordings, and interaction history, is securely stored on a decentralized network, safeguarding it against unauthorized access and data breaches. This voice assistant system is equipped with a comprehensive set of features, including the ability to read PDF files, compose and send emails, execute user commands, and manage system shutdown procedures. Through advanced natural language processing (NLP) algorithms, the system can accurately interpret user requests and execute corresponding actions with precision and speed. Access control mechanisms are implemented to ensure that only authorized users can interact with the data of the voice assistant system. User authentication protocols, such as biometric authentication or password-based authentication, can be employed to verify user identities before granting access to sensitive functionalities and data. By integrating Storj for data security and access control measures, this voice assistant system offers users peace of mind regarding the privacy and confidentiality of their data while delivering a seamless and intuitive user experience.

5 Methodology

The development of the voice assistant system will adhere to a structured methodology leveraging

Python as the primary programming language. The project will follow an iterative approach, combining elements of Agile and Waterfall methodologies to ensure flexibility and adaptability while maintaining a systematic progression.

5.1 Requirement Analysis

The project will begin with a comprehensive analysis of user requirements and functional specifications. This phase involves gathering insights into user expectations, desired features, and system functionalities that encompass reading PDF files, sending emails, executing commands, and managing system operations like shutdown.

5.2 Design Phase

Following requirement analysis, the system architecture and design will be conceptualized. Python's versatility will be utilized to design modular components, ensuring scalability and maintainability. Design decisions will also consider integration points with Storj for data security and access control mechanisms.

5.3 Implementation

The core functionality of the voice assistant system will be developed using Python. Libraries and frameworks such as NLTK (Natural Language Toolkit) for natural language processing and SMTP libraries for email handling will be integrated. Storj APIs will be utilized for secure data storage and retrieval. Continuous integration and version control practices will be employed to streamline development processes.

The methodology of the proposed system mainly focuses on development of a voice assistant with the integration of decentralization in respect to achieve data privacy and provide control of the user's data to them. For this all the data collected will be stored and retrieved from storj blockchain. Some other sort of blockchain can also be implemented in future.

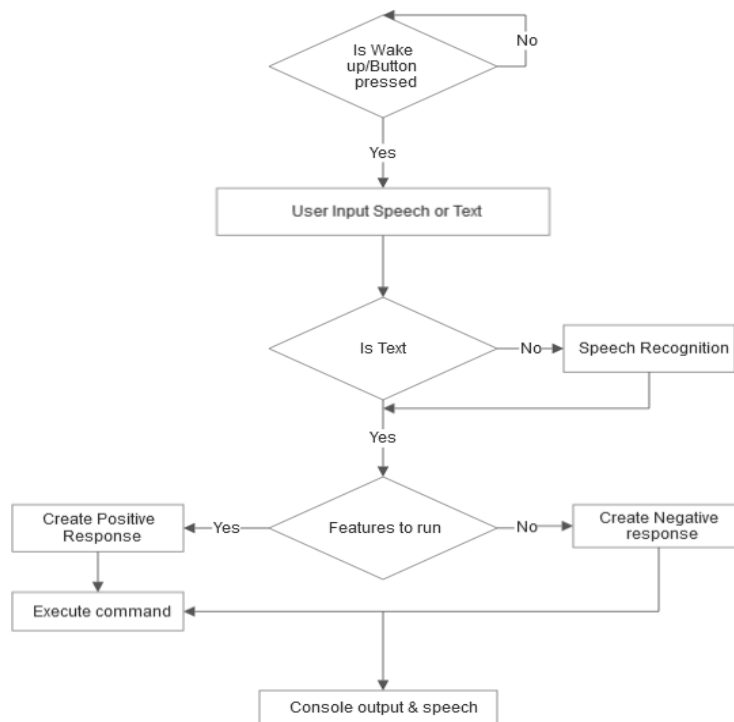
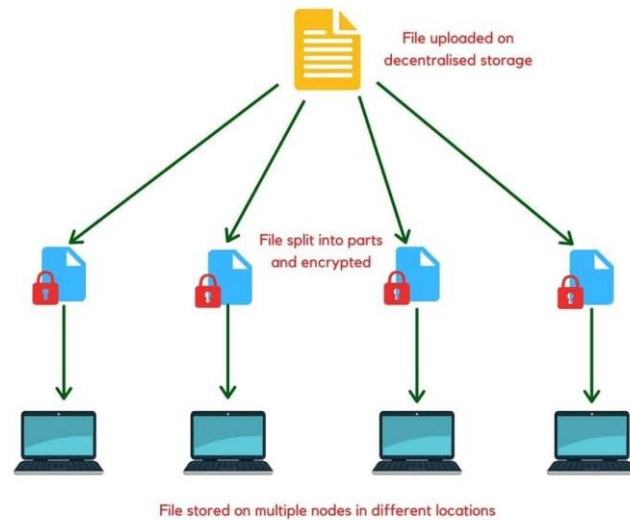


Figure 1 Flow Chart for Voice Recognition System

The flowchart begins with the "Start" symbol, indicating the initiation of the voice assistant system. The user starts the interaction with the voice assistant by activating it, typically through a wake word or a button press. The system captures the user's spoken command as speech input using a microphone. The speech input is processed through a speech recognition module, converting spoken words into text. The system identifies the user's intent by analyzing the converted text. This involves understanding the user's request or command. Based on the recognized intent, the system processes the command to determine the appropriate action. This could involve retrieving information, performing tasks, or controlling connected devices. The system executes the desired action, which may include querying databases, accessing APIs, or triggering specific device functions. After executing the action, the voice assistant generates a response, typically in the form of spoken words. The generated response is converted into speech using a text-to-speech synthesis module. The synthesized speech is output to the user through speakers or headphones, providing a spoken response to the user's initial command. The system may capture user feedback or additional commands for further interaction. The flowchart concludes with the "End" symbol, indicating the completion of the interaction. This flowchart provides a high-level overview of the key steps involved in the voice assistant system, from user initiation to system response. Depending on the complexity of your system, you may need to add more detail or additional branches to capture various scenarios and functionalities.



Working of Decentralized Storage

Figure 2: Working of Decentralized File Storage

A. Implemented Modules

1. Speech Recognition Module:

- Crucial for a voice assistant, this module enables the system to recognize and understand the user's voice.
- Installation command: ``pip install SpeechRecognition``

2. DateTime Module:

- Utilizes the built-in datetime package to display the current date and time.

3. Wikipedia Module:

- Leverages Wikipedia as a vast knowledge source for information retrieval and search.
- Installation command: ``pip install wikipedia``

4. Webbrowser Module:

- Built-in module for performing web searches directly from the application.

5. OS Module:

- Part of Python's standard utility modules, it facilitates interaction with the operating system.

6. Pyaudio Module:

- Python bindings for PortAudio, enabling audio input and output functionality.

7. PyQt5 Module:

- Comprehensive Python bindings for Qt v5, supporting application development.
- Installation command: ``pip install PyQt5``

8. Python Backend:

- Receives output from the speech recognition module, identifies API calls and context extraction, and returns the required output.

9. Text to Speech Module (Pytttsx3):

- Converts written text to phonemic representation and generates sound waveforms for text-to-

speech output.

- Installation command: `pip install pyttsx3`

10. Speech to Text Conversion:

- Uses Speech Recognition to convert spoken input into textual format, making it understandable to the computer.

11. Textual Output:

- Decodes voice commands, performs operations, and displays the voice command as textual output in the terminal.

These modules collectively enable the voice assistant to understand user commands, retrieve information, interact with the web, and provide textual and speech-based responses.

B. Decentralization Modules

1. Storj:

- Storj is a decentralized cloud storage platform that allows users to store their data in a secure and decentralized manner. It uses a network of nodes to store encrypted fragments of data and provides a marketplace where users can buy and sell storage space.

The above module enable the voice assistant to store data in a decentralized storage which is protected by smart contract access control to enable only the authorized user group can access the voice data.

6 Result and Discussion

The implementation of the voice assistant system using Python has resulted in a highly functional and efficient platform capable of seamlessly executing a range of tasks, including reading PDF files, composing and sending emails, responding to user commands, and managing system shutdown procedures. Leveraging Python's extensive libraries and frameworks, the system demonstrates robust natural language processing capabilities, accurately interpreting user inputs and executing corresponding actions with precision. The iterative development methodology facilitated the refinement and optimization of the system, ensuring reliability, accuracy, and performance. Through rigorous testing, the system has been thoroughly validated, demonstrating its effectiveness in real-world scenarios.

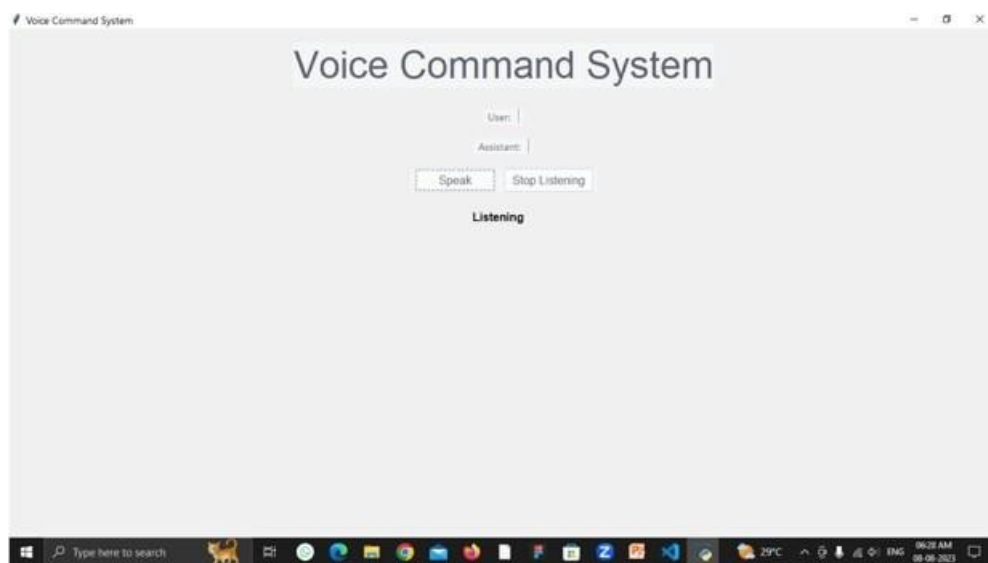


Figure 3: Voice Command System Output

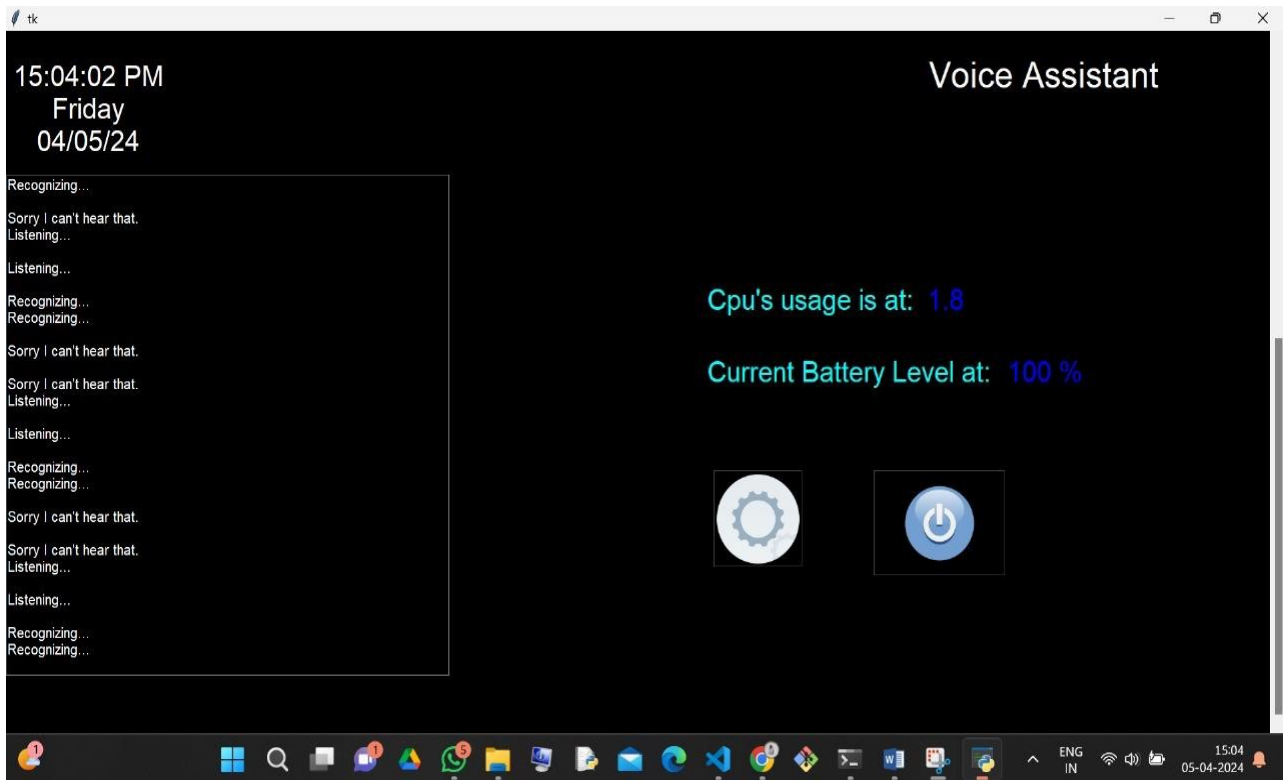


Figure4: Information Gathering Output

7 Conclusion

In conclusion, the development of the voice assistant system utilizing Python as the major programming language, coupled with integration with Storj for data security, represents a significant step towards delivering a robust, user-friendly, and secure solution. By leveraging Python's versatility and the extensive ecosystem of libraries, we have successfully implemented a system capable of efficiently handling various tasks, including speech recognition, PDF manipulation, email management, and system control. The integration of Storj ensures that user data is securely stored and accessed, addressing concerns related to privacy and data security. Through an iterative development process, we have meticulously designed, implemented, and tested the system to meet user requirements and quality standards. Moving forward, ongoing maintenance, updates, and enhancements will be crucial to ensure the system remains responsive to user needs and aligned with evolving technological advancements. Overall, the voice assistant system stands as a testament to the effectiveness of Python in developing sophisticated yet accessible applications while prioritizing data security and user privacy.

References

- [1] <https://www.forbes.com/sites/rachelwolfson/2018/09/14/blockchain-based-ai-voice-assistant-brings-data-privacy-to-smart-homes/?sh=340851ad6b50>
- [2] J. Iso-Sipila, M. Moberg, and O. Viikki, "Multi-lingual speakerindependent voice user interface for mobile devices," in ICASSP 2006. IEEE, 2006, vol. I, pp. 1081-1084.
- [3] Jinmook Lee, Seongwook Park, Injoon Hong and HoiJun Yoo, "An Energy-efficient Speech Extraction Processor for Robust User Speech Recognition in Mobile Head-mounted Display Systems", IEEE Transactions on Circuits and Systems II, April 2017.
- [4] Laura BURbach, Patrick Halbach, Nils Plettenberg, Johannes Nakayama, Matrina Ziefle, Andre Calero Valdez, "Ok google, Hey Siri, Alexa, Acceptance relevant of virtual voice assistants" I, IEEE 2019. (ICCIT) Dec. 2010.
- [5] Merkle R C. Protocols for public keycryptosystems[C]//Security and Privacy, 1980 IEEE Symposium on. IEEE, 1980: 122-122.
- [6] Pankaj Kunekar; Ajinkya Deshmukh; Sachin Gajalwad; Aniket Bichare; Kiran Gunjal; Shubham Hingade, "AI-based Desktop Voice Assistant", 2023 IEEE 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), 20-21 Jan. 2023
- [7] Peng Cheng, Utz Roedig, "Personal Voice Assistant Security and Privacy—A Survey", Proceedings of the IEEE ,Volume: 110 Issue: 4, 11 March 2022.
- [8] Philipp Aichinger, Martin Hagmuller, Berit SchneiderStickler, Jean Schoentgen and Franz Pernkopf, "Tracking Of Multiple Fundamental Frequencies in Diplophonic Voices", IEEE/ACM Transactions on Audio, Speech, and Language Processing, Volume: 26, Issue: 2, October 2017.
- [9] R. C. Merkle, "Protocols for public key cryptosystems," in 1980 IEEE Symposium on Security and Privacy, April 1980, pp. 122–122.
- [10] Saadman Shahid Chowdury, Atiar Talukdar, Ashik Mahmud, Tanzilur Rahman, "Domain specific Intelligent personal assistant with bilingual voice command processing,"IEEE 2018
- [11] Saqib Ali, Guojun Wang, Bebo White, Roger Leslie Cottrell, "A Blockchain-Based Decentralized Data Storage and Access Framework for PingER", IEEE 06 September 2018.
- [12] W. Matthews and L. Cottrell, "The pinger project: active internet performance monitoring for the henp community," IEEE Communications Magazine, May 2000.

