

AI EMPOWERED PERSONAL ASSISTANT SERVICE

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

KOLLI SRI NAGA PHANISH (Reg.No 40111259)

&

S.BHAVAN NAIDU (Reg.No - 40111238)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

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

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | 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 **KOLLI SRI NAGA PHANISH** (Reg.No-40111259) who carried out the Project Phase-2 entitled "AI EMPOWERED PERSONAL ASSISTANT SERVICE" under my supervision from November 2023 to April 2024.

Internal Guide

Dr.E,MURALI,M.Tech., Ph.D.,

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva-voce Examination held on

15-04-2024

Internal Examiner

15/4/2024
External Examiner

DECLARATION

I KOLLI SRI NAGA PHANISH(Reg. No - 40111259) hereby declare that the Project Phase- 2 Report entitled "A.I EMPOWERED PERSONAL ASSISTANT SERVICE" was done by me under the guidance of Dr.E.MURALI.,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

K. Phanish

SIGNATURE OF THE CANDIDATE

DECLARATION

I KOLLI SRI NAGA PHANISH(**Reg. No - 40111259**) hereby declare that the Project Phase- 2 Report entitled “**A.I EMPOWERED PERSONAL ASSISTANT SERVICE**” was done by me under the guidance of **Dr.E.MURALI.,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 :

PLACE : Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA** 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. E.MURALI.,M.Tech., Ph.D.**, for his valuable guidance, suggestions, and constant encouragement that paved way for the successful completion of my phase-2 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

This paper proposes the development of a voice assistant integrated into an augmented reality (AR) environment. Voice assistants have become increasingly popular in recent years, with the widespread adoption of smart speakers and mobile devices. Meanwhile, AI technology has also advanced rapidly, offering new ways to interact with digital information and objects in the real world.

The proposed system aims to combine the convenience and ease of use of a voice assistant with the immersive and interactive nature of AR. The user will be able to access the voice assistant through a wearable device, such as smart glasses or a headset. The voice assistant will respond to the user's voice commands and provide relevant information and assistance, displayed as virtual overlays in the AR environment.

The system will use natural language processing and machine learning algorithms to interpret the user's voice commands and provide accurate and helpful responses. The voice assistant will also be able to learn from the user's interactions and adapt to their preferences and habits over time.

The potential applications of the system are numerous, ranging from personal assistance to industrial and commercial use cases. For example, a user could use the voice assistant to navigate through a city or a museum, receive real-time translations, or access information about products in a store. In a factory or a warehouse, workers could use the system to receive instructions and guidance, improving their efficiency and safety.

The implementation of the system will require overcoming technical challenges related to the integration of voice recognition and AR technology, as well as ensuring user privacy and security. However, the potential benefits of a voice assistant in an AR environment make it a promising area for future research and development.

TABLE FOR CONTENTS

CHAPTER No	TITLE	PAGE No
	ABSTRACT	v
	LIST OF FIGURE	viii
	LIST OF TABLES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
	2.1 Inferences from Literature Survey	2
	2.2 Open problems in Existing System	6
3	REQUIREMENTS ANALYSIS	9
	3.1 Feasibility Studies/Risk Analysis of the Project	9
	3.2 Software Requirements Specification Document	11
	3.2.1 Windows 10 or above	12
	3.2.2 Unity 5	12
	3.2.3 PyCharm	14
	3.2.4 Python	14
	3.2.5 AR Core	16
	3.2.6 Visual Studio	17
4	DESCRIPTION OF PROPOSED SYSTEM	18
	4.1 Selected Methodology or process model	19
	4.1.1 Permissions needed	20
	4.2 Architecture / Overall Design of Proposed System	21
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	25
	4.3.1 System Testing	25

	4.3.2	Unit Testing	26
	4.3.3	Integration Testing	27
	4.4	Project Management Plan	27
5		IMPLEMENTATION DETAILS	29
	5.1	Development and Deployment setup	29
	5.2	Algorithms	30
6		RESULTS AND DISCUSSION	32
7		CONCLUSION	34
	7.1	Conclusion	34
	7.2	Future Work	34
	7.3	Research Issues	35
	7.4	Implementation	36
		REFERENCES	38
		APPENDIX	
		A.SOURCE CODE	39
		B.SCREENSHOTS	55
		C.RESEARCH PAPER	56

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	Features of Natural Language Processing	15
4.1	Workflow of System	22
4.2	Workflow Flowchart of our proposed model	23

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
4.1	PROJECT MANAGEMENT PLAN	27

LIST OF ABBREVIATIONS

SHORT FORM

FULL FORM

AR	AUGMENTED REALITY
ARVA	AUGMENTED REALITY VOICE ASSISTANCE
API	APPLICATION PROGRAMMING INTERFACE
HMM	HIDDEN MARKOV MODEL
NLP	NATURAL LANGUAGE PROCESSING
SDK	SOFTWARE DEVELOPMENT KIT
TTS	TEXT TO SPEECH
VS	VISUAL STUDIO
VR	VIRTUAL REALITY

CHAPTER 1

INTRODUCTION

Voice assistants have become an integral part of our daily lives, with devices like Amazon's Alexa, Apple's Siri, and Google Assistant providing convenient access to information and assistance. Meanwhile, augmented reality (AR) technology has also advanced rapidly, allowing us to interact with digital information and objects in the real world.

In this context, the integration of voice assistants with AR technology presents an exciting opportunity to create a more immersive and interactive experience for users. A voice assistant in an AR environment would allow users to access information and assistance through natural language voice commands, while also displaying relevant visual information in the AR space.

The potential applications of such a system are numerous. For example, a user could use the voice assistant to navigate through a city or a museum, receive real-time translations, or access information about products in a store. In industrial or commercial settings, workers could use the system to receive instructions and guidance, improving their efficiency and safety.

Implementing such a system would require overcoming technical challenges related to the integration of voice recognition and AR technology, as well as ensuring user privacy and security. However, the benefits of a voice assistant in an AR environment make it a promising area for research and development.

This paper proposes the development of a voice assistant integrated into an AR environment. The system will use natural language processing and machine learning algorithms to interpret user voice commands and provide relevant information and assistance.

CHAPTER 2

LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY

The integration of voice assistants and augmented reality (AR) technology presents an exciting opportunity for creating a more immersive and interactive experience for users. Voice assistants are becoming increasingly popular, with devices such as Amazon's Alexa, Apple's Siri, and Google Assistant providing convenient access to information and assistance. Meanwhile, AR technology has advanced rapidly, allowing us to interact with digital information and objects in the real world. This literature survey aims to explore the current state of research on the integration of voice assistants and AR technology, the potential applications of such a system, and the challenges that need to be addressed.

Current State of Research: Research on the integration of voice assistants and AR technology is still in its early stages. However, there have been some notable developments in recent years. For example, researchers at MIT have developed an AR system called "AlterEgo" that allows users to control digital devices through voice commands and subtle facial gestures, without speaking out loud. The system uses a wearable device that measures muscle activity in the face and throat to interpret the user's commands. Another example is the "HoloLens" device developed by Microsoft, which combines AR and voice commands to create a more intuitive and natural user interface. In addition to academic research, there are also several companies working on the integration of voice assistants and AR technology. For example, the company "ThirdEye" has developed an AR headset that allows users to interact with a voice assistant and receive real-time translations, among other features. Similarly, the company "Vuzix" has developed a smart glasses device that integrates with Amazon's Alexa voice assistant.

Potential Applications:

The potential applications of a voice assistant in an AR environment are numerous.

One of the most obvious is navigation. Users could use the system to navigate through a city or a museum, with the voice assistant providing directions and relevant information. Another potential application is language translation. Users could speak to the voice assistant in their native

language, and the system could provide real-time translations in the AR environment. This could be particularly useful for international travel or business.

In industrial or commercial settings, there are also numerous potential applications. Workers could use the system to receive instructions and guidance, improving their efficiency and safety. For example, a worker in a factory could use the voice assistant to receive instructions on how to perform a particular task, with the instructions displayed as virtual overlays in the AR environment.

There are several technical and practical challenges that need to be addressed to successfully integrate voice assistants and AR technology. One of the main technical challenges is the integration of voice recognition and AR technology. Voice recognition technology relies on clear and accurate speech input, which can be difficult in noisy or crowded environments. In addition, the accuracy of voice recognition algorithms can be affected by accents and dialects.

Another challenge is the need to ensure user privacy and security. Voice assistants and AR devices often collect data on user behavior and preferences, which can be sensitive information. It is important to ensure that this data is protected and that users have control over how their data is used.

the integration of voice assistants and AR technology presents a promising area for research and development. The potential applications of such a system are numerous, ranging from personal assistance to industrial and commercial use cases. However, there are also several technical and practical challenges that need to be addressed, including the integration of voice recognition and AR technology, and ensuring user privacy and security. Further research is needed to address these challenges and to fully explore the potential of a voice assistant in an AR environment.

In recent years, there have been some significant developments in the integration of voice assistants and AR technology. For example, Google has developed an AR platform called "ARCore" that allows developers to create AR applications that incorporate voice commands. The platform uses Google's speech recognition technology to interpret voice commands and provide relevant information and assistance.

Another example is the "Magic Leap One" headset developed by Magic Leap, which combines AR and voice commands to create a more intuitive and natural user interface. The device uses spatial audio technology to provide audio feedback that is spatially aligned with the virtual objects in the AR environment, creating a more immersive experience for the user.

One of the potential challenges of integrating voice assistants and AR technology is the potential for information overload. AR devices can display a large amount of information in the user's field of view, and combining this with voice commands could potentially lead to a cluttered and confusing user experience. It is important to design the system in a way that provides relevant information without overwhelming the user.

Another challenge is the need to ensure that the system is accessible to users with different abilities and preferences. For example, users with hearing impairments may have difficulty with voice commands, while users with visual impairments may have

difficulty with visual overlays in the AR environment. It is important to design the system in a way that accommodates different user needs and preferences.

1. **Hong et al. (2019):** examined the use of ARVA in the education domain. The authors developed an ARVA based mobile app that allowed students to interact with virtual learning materials in a more engaging and interactive way. The results showed that the ARVA-based app significantly improved student motivation and learning outcomes.
2. **Cerratto-Pargman and Fernaeus (2019):** discussed the challenges of designing ARVA for educational purposes. The authors highlighted the need for a more user-centered design approach, as well as the importance of considering the socio- cultural context in which ARVA is being used.
3. **Xiang et al. (2019):** investigated the challenges of integrating virtual assistant functionality with AR technology. The authors identified
4. **Martínez-García et al. (2021):** discussed the challenges of protecting user privacy and securing data in ARVA. The authors highlighted the importance of implementing appropriate data protection measures and informing users about the collection and use of their data
5. **Prabhu et al. (2018):** healthcare industry, ARVA has been explored as a tool for improving patient outcomes and education. found that ARVA-based patient education significantly improved patient knowledge and satisfaction.

6. **Lee and Han (2019):** ARVA has also been used in the tourism industry to enhance the visitor experience. found that ARVA-based tour guides improved visit or engagement and satisfaction.
7. **Chai et al. (2021):** The use of ARVA in remote collaboration has been explored as well. found that ARVA-based remote collaboration improved communication and task performance for distributed teams.

The use of ARVA in gaming has also been explored, with games such as PokemonGo and Ingress using AR to provide an immersive gaming experience. The success of these games has led to increased interest in ARVA as a platform for gaming and entertainment. ARVA has been used for employee training in various industries
8. **Zhang et al. (2020):** The learning with ARVA has been explored as well. This integration allows ARVA to provide more personalized and context-aware assistance to users. (2020) found that the use of machine learning in ARVA improved user satisfaction and task performance.

Overall, the literature suggests that ARVA has significant potential in various domains, including retail, education, and hospitality. However, technical challenges, user adoption, and privacy concerns need to be addressed for successful implementation.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

The integration of voice assistants and augmented reality (AR) technology has the potential to create more intuitive and natural user interfaces for a variety of applications, ranging from personal assistance to industrial and commercial use cases. However, there are several open problems that need to be addressed to fully realize the potential of this technology. These problems include the accuracy and

robustness of voice recognition, privacy and security concerns, designing intuitive and natural user interfaces, accommodating different user needs and preferences, overcoming technical limitations, and integrating with existing systems. Addressing these open problems is essential to create a more robust, secure, and user-friendly voice assistant system in an AR environment. In this section, we will explore each of these open problems in more detail

Accuracy and robustness of voice recognition: One of the primary challenges in integrating voice assistants and AR technology is the accuracy and robustness of voice recognition in noisy environments. AR devices are often used in noisy environments such as construction sites or factory floors, where there is a lot of ambient noise. Voice recognition algorithms need to be robust enough to recognize voice commands accurately in such environments.

The factors for these difficulties are:

1. ***Privacy and security:*** AR devices have the potential to capture sensitive information such as personal conversations or industrial secrets. It is important to ensure that the voice assistant system is designed in a way that protects user privacy and data security.
2. ***Designing intuitive and natural user interfaces:*** Voice assistants and AR technology have the potential to create more intuitive and natural user interfaces. However, designing such interfaces requires careful consideration of factors such as the user's physical environment, the user's task and goals, and the user's preferences and abilities.
3. ***Accommodating different user needs and preferences:*** AR devices are used by users with different needs and preferences, such as users with disabilities, users who speak different languages, or users with different cultural back-

grounds. It is important to design the voice assistant system in a way that accommodates different user needs and preferences.

4. ***Overcoming technical limitations:*** AR devices are often limited by factors such as battery life, processing power, and connectivity. Voice assistants need to be designed in a way that takes into account these technical limitations to ensure optimal performance.
5. ***Integrating with existing systems:*** Voice assistant systems need to be integrated with existing systems such as enterprise resource planning (ERP) systems, customer relationship management (CRM) systems.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES / RISK ANALYSIS OF THE PROJECT

As with any new technology, the integration of voice assistants and augmented reality (AR) brings with it a range of potential risks that must be carefully analyzed and addressed. In this section, we will explore the various risks associated with the use of voice assistants in an AR environment and their potential impact on users, organizations, and society as a whole.

Privacy and security risks are among the most critical risks associated with voice assistants in AR environments. AR devices have the potential to capture sensitive information such as personal conversations or industrial secrets, raising concerns about data privacy and security. In addition to privacy and security risks, accuracy and reliability issues can also pose significant risks to users and organizations. If the voice assistant system is not accurate or reliable, it could lead to errors and mistakes, potentially causing harm to the user or the organization. also present risks to the functionality of voice assistants in AR environments. Finally, legal and ethical risks such as data privacy, user consent, and liability for errors or mistakes must be addressed to ensure compliance with legal and ethical standards. by carefully analyzing and addressing these risks, organizations can create a safe and effective voice assistant system in an AR environment, providing users with an intuitive and natural user interface while protecting their privacy and data security.

1. ***Privacy and Security Risks:*** The integration of voice assistants and AR technology can raise concerns about data privacy and security. AR devices have the potential to capture sensitive information such as personal

conversations or industrial secrets. The voice assistant system needs to be designed in a way that protects user privacy and data security. If the system is hacked, user data could be compromised, leading to financial losses, identity theft, or other negative consequences.

2. ***Accuracy and Reliability Risks:*** The accuracy and reliability of the voice assistant system can also pose risks to the user and the organization. If the system is not accurate or reliable, it could lead to errors and mistakes, potentially causing harm to the user or the organization. For example, if a voice command is misinterpreted in a manufacturing plant, it could lead to a production error or even a safety hazard
3. ***User Experience Risks:*** The user experience is also a critical risk factor for voice assistants in AR environments. If the system is not designed to be intuitive and user-friendly, it could lead to user frustration or even user error. If the user interface is too complicated or difficult to use, it could lead to reduced productivity and efficiency
4. ***Technical Limitations Risks:*** AR devices are often limited by factors such as battery life, processing power, and connectivity. Voice assistants need to be designed in a way that takes into account these technical limitations to ensure optimal performance. If the system is not designed to work within the technical limitations of the AR device, it could lead to poor performance and reduced user satisfaction.
5. ***Legal and Ethical Risks:*** The integration of voice assistants and AR technology raises legal and ethical concerns, such as data privacy, user consent, and liability for errors or mistakes. Organizations need to ensure that they comply with legal and ethical standards and that they are transparent about the use of voice assistant technology in AR environments.
6. ***Social Risks:*** The use of voice assistants in AR environments could also have social implications. For example, the use of voice assistants in public spaces could lead to privacy concerns for nearby individuals who may inadvertently

overhear private conversations. Additionally, voice assistants could lead to a reduction in human-to-human interaction, potentially leading to social isolation and reduced social skills.

7. **Bias Risks:** Voice assistants are trained using machine learning algorithms, which can be biased based on the data used to train them. If the training data is biased, it could lead to biased results and potentially discriminatory outcomes. This is particularly important to consider in AR environments where voice assistants are used for decision-making processes such as hiring or employee evaluations
8. **Interoperability Risks:** Interoperability between different AR devices and voice assistant systems is a significant challenge. As different vendors may use different technologies and protocols, it can be difficult to ensure that different devices and systems work seamlessly together. This could lead to user frustration and reduced productivity.
9. **Maintenance and Upkeep Risks:** AR devices and voice assistants require regular maintenance and upkeep to ensure optimal performance. Organizations need to consider the costs associated with maintaining and updating the system over time. Failure to do so could lead to reduced performance, increased downtime, and potential security risks.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT:

- Windows 10 or above
- Unity 5
- PyCharm
- Python

-
- Augmented Reality SDK (AR Foundation)
- Visual Studio

3.2.1 Windows 10 or above

Windows 10 is the latest operating system developed and released by Microsoft. It was first introduced in 2015 as a free upgrade for users of Windows 7 and 8.1. Since then, it has become one of the most widely used operating systems worldwide, thanks to its improved user interface, enhanced security features, and improved performance. One of the key features of Windows 10 is its user interface. Microsoft has taken a minimalist approach with the design,

Since then, it has become one of the most widely used operating systems worldwide, thanks to its improved user interface, enhanced security features, and improved performance. One of the key features of Windows 10 is its user interface. Microsoft has taken a minimalist approach with the design, making it much cleaner and more intuitive compared to previous versions of Windows. The Start menu has returned, and users can customize it with their favorite apps and programs. Additionally, Windows 10 features a virtual assistant called Cortana, which allows users to search the web, set reminders, and perform other tasks using voice commands.

3.2.2 Unity 5

Unity is a cross-platform game engine that is widely used by game developers to create high-quality video games and interactive experiences for various platforms such as Windows, Mac, iOS, Android, and many others. Unity is a powerful tool that enables developers to create 2D and 3D games, simulations, and other interactive applications with ease. The engine is designed to be intuitive and easy to use, making it an ideal choice for developers of all skill levels.

One of the key strengths of Unity is its ability to create games that are visually stunning and highly immersive. With advanced rendering techniques such as dynamic lighting, real-time shadows, and global illumination, Unity allows developers to create highly detailed and realistic environments that transport

players into immersive game worlds. Additionally, Unity's robust physics engine allows developers to create complex and realistic simulations, including advanced character animations, collision detection, and ragdoll physics.

Another strength of Unity is its ability to handle complex game logic and scripting. Unity supports several programming languages, including C# and JavaScript, which allows developers to create custom game mechanics, AI, and other advanced features. The engine also includes a visual scripting tool called "Unity Playmaker," which enables developers to create complex game logic without writing code.

Unity is also highly versatile, making it an ideal choice for game development studios of all sizes. The engine supports a wide range of platforms and devices, including mobile, desktop, and console gaming platforms. Additionally, Unity has a large and active community of developers who create and share tools and assets that can be used to enhance the development process.

In recent years, Unity has also become increasingly popular for non-game applications such as virtual and augmented reality (VR/AR) experiences. The engine's support for VR/AR devices and its ability to create highly immersive environments make it an ideal tool for creating cutting-edge experiences in these emerging technologies.

Unity is a powerful game engine that offers a wide range of features and tools for game development. Its strengths lie in its ability to create visually stunning and highly immersive environments, its robust physics engine, and its ability to handle complex game logic and scripting. With its versatility and large community of developers, Unity has become a go-to choice for game developers and studios of all sizes.

3.2.3 PyCharm

PyCharm is an integrated development environment (IDE) specifically designed for Python programming. Developed by JetBrains, PyCharm provides a range of features and tools that make it easier for developers to write, debug, and deploy Python applications.

One of the key features of PyCharm is its code editor. The editor provides advanced code completion, syntax highlighting, and error highlighting to help developers write clean and error-free code. Additionally, PyCharm includes features such as code refactoring, debugging tools, and version control integration, which help developers to work more efficiently and collaborate more effectively.

PyCharm also includes a range of tools for testing and debugging Python code. It provides a built-in debugger that allows developers to step through code and identify and fix errors quickly. Additionally, PyCharm includes tools for unit testing and integration testing, as well as support for popular testing frameworks such as pytest and unittest. It also includes support for popular cloud platforms such as Microsoft Azure, and Google Cloud Platform.

Another notable feature of PyCharm is its support for multiple frameworks and libraries. PyCharm supports a wide range of Python frameworks, such as Django, Flask, Pyramid, and others. It also includes a built-in package manager and an extensive library of plugins that help developers to extend and customize.

3.2.4 Python

One of the strengths of Python is its simplicity and ease of use. The language is designed to be easy to read and write, with a syntax that is straightforward and intuitive. This makes it an ideal language for beginners, as well as for experienced developers who want to work quickly and efficiently.

Python is also known for its vast libraries and frameworks, which provide a wide range of tools and resources for developers. Some popular libraries and

frameworks include NumPy, Pandas, Django, Flask, TensorFlow, and PyTorch, among many others.

Speech Recognition and Natural Language Processing: For a voice assistant, speech recognition and NLP libraries are essential. Popular libraries include Google Cloud Speech-to-Text API, Microsoft Azure Speech Services, and Amazon Transcribe.

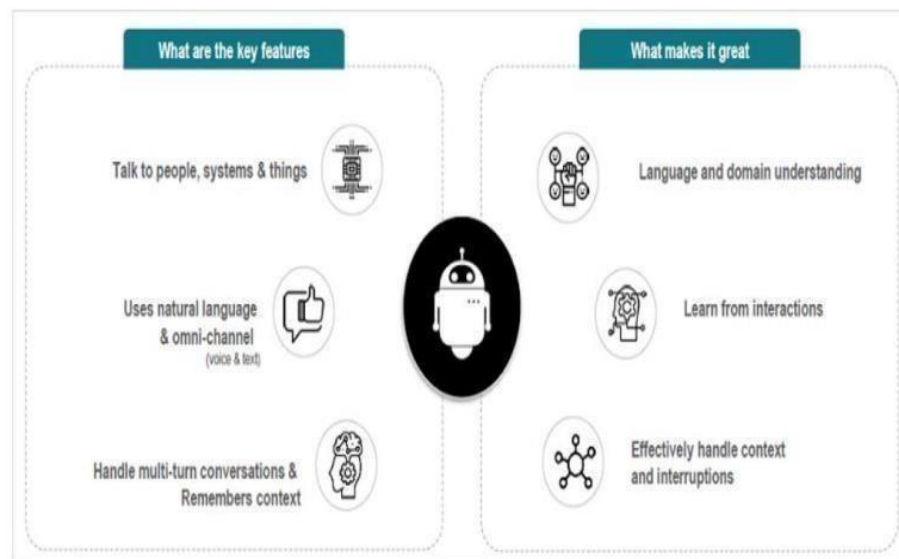


Fig 3.1 Features of Natural Language Processing

Text-to-Speech (TTS): TTS libraries are needed to convert the responses of the voice assistant to speech. Some popular TTS libraries are Google Cloud Text-to-Speech API, Amazon Polly, and Microsoft Azure Speech Services.

Wolfram Alpha API: Wolfram Alpha is a computational knowledge engine developed by Wolfram Research. It is designed to answer factual questions by computing the answers from structured data sources and a vast collection of algorithms. Wolfram Alpha is different from traditional search engines like Google

in that it computes answers to questions rather than simply providing links to web pages

Pyjokes: Pyjokes is a Python library that provides a collection of funny and humorous jokes that can be easily incorporated into Python code. It is a simple and lightweight library that can add a touch of humor to Python scripts and projects.

The library contains a variety of jokes related to programming, coding, and technology. It can be used in a variety of ways, such as in console applications, web applications, or even in chatbots. Pyjokes is an open-source library and is available for free on GitHub.

Open Weather API: OpenWeather API is a web-based service that provides weather data for developers to integrate into their applications. The API provides real-time weather data, historical data, and 5-day forecast data for any location in the world.

Developers can access the OpenWeather API using a RESTful interface, and can retrieve data in a variety of formats, including JSON, XML, and HTML. The API provides a variety of weather data, including temperature, humidity, wind speed and direction, cloud cover, and precipitation

3.2.5 AR Core

ARCore is a software development kit (SDK) developed by Google that allows developers to create augmented reality (AR) experiences for Android devices. ARCore uses advanced computer vision technologies to detect the position, orientation, and lighting of the user's phone, and to anchor virtual objects to the real world.

ARCore, developers can create AR experiences that are interactive, immersive, and realistic. The SDK includes tools and APIs for motion tracking, environmental understanding, and light estimation, allowing developers to build AR applications that can respond to the user's movements and the environment around them.

3.2.6 Visual Studio

Visual Studio is a popular integrated development environment (IDE) developed by Microsoft. It is used primarily for developing software applications, websites, and mobile apps for Windows, Android, iOS, and other platforms. The IDE supports a wide range of programming languages, including C#, C++, Visual Basic, Python, and JavaScript

Visual Studio includes a variety of tools and features to streamline the development process, including a code editor with syntax highlighting, code completion, and debugging capabilities. It also includes a visual design interface for building user interfaces and graphical components, as well as support for version control systems such as Git and TFS

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

The proposed system for the Voice Assistant in Augmented Reality is a software application that combines the capabilities of a voice assistant with the immersive experience of augmented reality technology. The system is designed to provide users with a more intuitive and natural way to interact with technology, by enabling them to use voice commands to control and interact with virtual objects in an augmented reality environment.

The system will consist of a mobile application that can be installed on a smartphone or tablet, which will use the device's camera and sensors to create an augmented reality environment. The application will also include a voice assistant feature, which will allow users to give voice commands to control and interact with virtual objects in the augmented reality environment.

The system will also include a range of other features, such as the ability to set reminders, create to-do lists, and control smart home devices. The user will be able to activate these features using voice commands, and the system will respond with relevant actions.

1. **Intended use:** Provide users with a more intuitive and natural way to interact with technology.
2. **Potential impact:** Revolutionize the way we interact with technology by making it more natural, intuitive, and immersive
3. **Technology:** Mobile device camera and sensors, natural language processing technology, augmented reality technology
4. **Use cases:** Weather forecasting, navigation, productivity, smart home control, entertainment, education

5. **Target audience:** Users who value convenience, efficiency, and immersive experiences, particularly those who are comfortable with voice-activated assistants and mobile technology
6. **Competitors:** Other voice assistants, augmented reality applications, mobile productivity and entertainment apps
7. **Challenges:** Integration of voice assistant and augmented reality technologies, accuracy of natural language processing, potential privacy concerns
8. **Opportunities:** Expanding use cases for augmented reality and voice assistants, potential for personalized and customizable experiences, potential for integration with other technologies and services

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

Methodology for developing a voice assistant in augmented reality could be an iterative, user-centered design approach. This could involve the following steps:

1. **User research:** Conduct research to understand the needs, preferences, and pain points of potential users, as well as their familiarity with voice assistants and augmented reality technology.
2. **Conceptualization:** Use the insights from user research to develop initial concepts and use cases for the voice assistant in augmented reality, taking into account potential technical limitations and opportunities.
3. **Prototyping:** Create prototypes of the voice assistant in augmented reality, which could involve developing wireframes, low-fidelity mockups, or working prototypes using tools like Unity or ARCore.
4. **User testing:** Conduct usability testing with potential users to gather feedback and insights on the user experience, including the effectiveness of the voice assistant and augmented reality features.

5. **Iteration:** Use the insights gathered from user testing to refine and improve the design of the voice assistant in augmented reality, potentially involving multiple rounds of testing and refinement.
6. **Implementation:** Develop the final version of the voice assistant in augmented reality, incorporating the insights and feedback gathered during the user-centered design process.
7. **Launch and evaluation:** Launch the voice assistant in augmented reality and evaluate its effectiveness and user adoption, potentially incorporating feedback from users to continue to refine and improve the experience over time

4.1.1 Permissions needed

The permissions needed for a voice assistant in augmented reality will depend on the specific features and functionalities of the application. Some potential permissions that may be required include:

1. **Camera access:** If the voice assistant is being used in conjunction with augmented reality technology, it may also require permission to access the device's camera.
2. **Location access:** If the voice assistant includes location-based features, it may need permission to access the device's GPS or other location services.
3. **Internet access:** The voice assistant may need permission to access the internet in order to perform searches or retrieve information.
4. **Contacts access:** If the voice assistant includes contact-based features, such as making phone calls or sending text messages, it may need permission to access the device's contact list.
5. **Storage access:** The voice assistant may need permission to access the device's storage in order to store user preferences, data, or other information.
6. **Bluetooth Access:** If the voice assistant is integrated with other Bluetooth-enabled devices, it may require access to the device's Bluetooth functionality.

7. *Calendar Access*: If the voice assistant is integrated with the device's calendar, it may require access to the calendar data to schedule appointments or set reminders.
8. *Sensor Access(optional)*: In addition to camera and microphone access, the voice assistant may require access to other sensors on the device, such as the gyroscope or accelerometer, to track and analyze user movements.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

The architecture of a voice assistant in augmented reality would likely include the following components

1. ***Augmented Reality Framework***: This component is responsible for rendering virtual objects into the real-world environment, based on the device's camera input.
2. ***Speech Recognition Engine***: This component listens to the user's voice commands and transcribes them into text, which is then sent to the natural language processing component.
3. ***Natural Language Processing (NLP)***: This component processes the user's text input, identifying the user's intent and extracting relevant information. The NLP component uses machine learning algorithms and language models to understand the user's input and generate a response.
4. ***Dialogue Management***: This component determines the appropriate response to the user's input based on the context of the conversation and the user's intent. The dialogue management component uses decision trees or rule-based systems to determine the appropriate response.

5. **Text-to-Speech (TTS):** This component converts the response generated by the dialogue management component into spoken language, which is then played through the device's speakers.
6. **Application Programming Interfaces (APIs):** This component provides access to external services, such as weather forecasts or news updates, which can be integrated into the voice assistant's functionality.
7. **User Profile Management:** This component allows the voice assistant to store user preferences and settings, such as preferred language, preferred news sources, or frequently visited locations. The user profile management component ensures that the voice assistant can provide a personalized experience for each user.

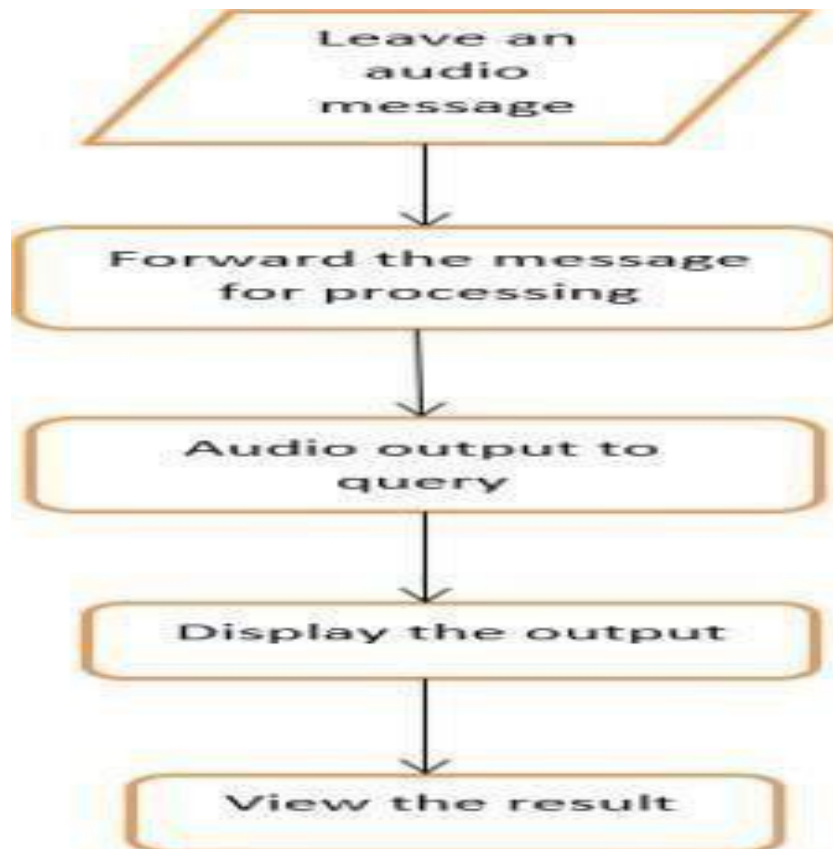


FIGURE 4.1 Workflow of system

When user leaves an audio message, the message is forwarded for process and audio output to query is displayed on the screen

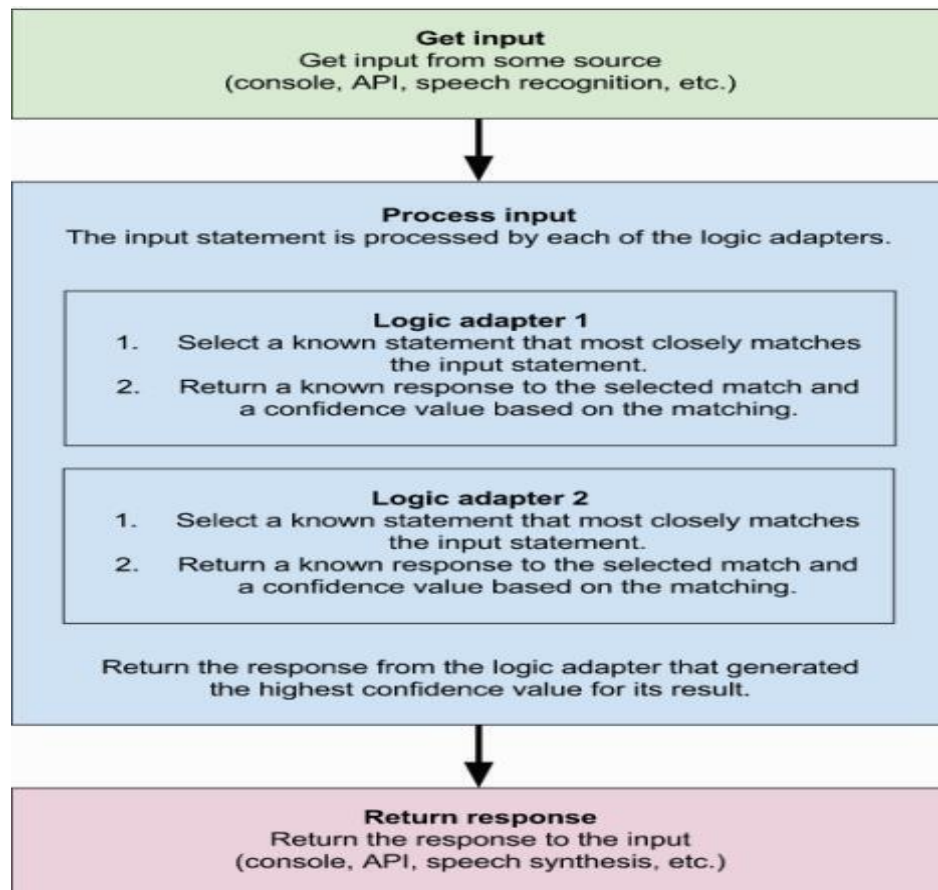


Fig:4.2 Flowchart of our proposed model

Our proposed model for a chatbot is designed to revolutionize the way humans interact with technology. We aim to create a chatbot that can interact with humans in a way that mimics a human-to-human conversation, with the help of technologies like AI natural language processing, augmented reality, and robust network connectivity.

One of the main advantages of our proposed model is that it can take input from the user in the form of an audio message. This means that users do not have to type out their queries, which can be time-consuming and inconvenient. Instead, they can simply speak their query out loud, and the chatbot will process it.

To ensure that the audio message is processed accurately, we have included a voice messenger in our proposed model. This messenger is responsible for processing the audio message and passing it on to the API chatbot. This step ensures that the chatbot receives the correct input from the user and can provide an accurate response.

The AI natural language processing technology used in our proposed model is another key feature that makes it stand out from traditional chatbots. This technology enables the chatbot to understand the user's query and respond in a way that is natural and intuitive. The chatbot can extract entities and intents from the user's message, enabling it to provide a more personalized response.

In addition to AI natural language processing, we have also incorporated augmented reality into our proposed model. Augmented reality technology allows the chatbot to interact with the user in a more immersive and engaging way. For example, the chatbot could use AR to display information to the user in a way that feels like it is part of their environment.

Despite the many advantages of our proposed model, there are some limitations that need to be considered. One such limitation is that personalized answers may not be possible in all cases. While our chatbot can extract entities and intents from the user's message, it may not always be able to provide a completely personalized response. This is because some queries may be too complex or unique for the chatbot to understand fully.

Another limitation to consider is that standards regarding augmented reality app design and development are still relatively limited. While AR technology is becoming more widely used, there are still many best practices and guidelines that need to be developed to ensure that AR experiences are user-friendly and engaging. Overall, our proposed model for a chatbot that uses AI natural language processing, augmented reality, is an exciting development in the world of conversational AI. By creating a chatbot that can interact with users in a more natural and intuitive way, we hope to make technology more accessible and user-friendly for everyone. While there are limitations to be aware of, we believe that the benefits of our proposed model far outweigh any potential drawbacks. We look forward to seeing how this technology develops in the future and how it can be used to enhance the way we interact with technology.

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

4.3.1 System Testing

System testing is a level of software testing where the whole system is tested as per the requirements specified in the Software Requirement Specification (SRS) document. It is conducted after the completion of integration testing and is used to evaluate the quality and performance of the entire system. The main objective of system testing is to verify that the developed system meets all the functional and non-functional requirements and works as expected.

In system testing, the system is tested in its entirety, including all its components and modules. This includes testing the user interface, integration with other systems, data processing, security, performance, and reliability. System testing

is usually performed by a dedicated team of testers who have not been involved in the development process.

There are different types of system testing, such as functional testing, performance testing, stress testing, security testing, and usability testing. The type of testing depends on the requirements specified in the SRS document and the goals of the project.

System testing is a critical phase of the software development life cycle, as it ensures that the developed system meets all the requirements and is ready for deployment. It helps to identify any defects or issues in the system and ensures that they are fixed before the system is deployed in a live environment.

4.3.2 Unit testing

Unit testing is a type of software testing in which individual units or components of a software application are tested in isolation to verify that they are working as expected. The goal of unit testing is to isolate and test each part of the application code to ensure that it is functioning correctly.

Unit testing is usually performed by developers and is conducted during the development phase of the software development life cycle. It helps to identify bugs and errors early in the development process, which reduces the cost of fixing them later on. It also helps to improve the overall quality of the software and ensures that it meets the specified requirements.

In unit testing, each unit or component of the application is tested independently of the other components. This means that the dependencies on other components are usually mocked or stubbed to isolate the unit being tested. Unit tests are typically automated and can be run repeatedly to ensure that the code changes do not introduce any new bugs.

Unit testing frameworks and tools such as JUnit, NUnit, and PyUnit provide support for writing and executing unit tests. These frameworks make it easier to write test cases, execute tests, and report the results.

4.3.3 Integration testing

Integration testing is a type of software testing that is performed to test the integration between different components or modules of a software application. The main goal of integration testing is to verify that the individual modules or components that have been developed separately work correctly when integrated together as a larger system.

Integration testing is usually performed after unit testing, and it can be done at different levels of the software hierarchy, such as component integration, system integration, and acceptance integration. In component integration testing, the focus is on testing the integration between individual modules or components of the software application. In system integration testing, the focus is on testing the interaction and communication between different subsystems of the application. In acceptance integration testing, the focus is on testing the entire system in a production-like environment.

Integration testing can be performed using different strategies, such as top-down, bottom-up, and hybrid approaches. In the top-down approach, testing starts with the higher-level modules or components, and then the lower-level modules are gradually integrated and tested. In the bottom-up approach, testing starts with the lower-level modules, and then the higher-level modules are gradually integrated and tested. The hybrid approach combines elements of both top-down and bottom-up strategies.

4.4 PROJECT MANAGEMENT PLAN

Table 4.1 Process time

Add API	30/11/2022
---------	------------

Debugging(Python)	02/08/2023
Character creation(AR)	31/08/2023
Scene creation(Unity)	10/09/2023
Scripting(C#)	30/09/2023
Debugging	01/10/2023

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP :

The voice assistant in augmented reality system will involve the use of several tools and technologies.

Firstly, we will use Unity 3D game engine to create the augmented reality environment, including the user interface, interactive elements, and the virtual voice assistant character. We will also use ARCore for tracking the user's environment and positioning the virtual objects in the correct location.

For the voice recognition and natural language processing, we will use Python programming language and the Speech Recognition library to capture the user's voice input and convert it to text. We will also use the Natural Language Toolkit (NLTK) library for processing and understanding the user's input.

To provide the voice output, we will use the Text-to-Speech (TTS) engine, which can generate human-like voice output based on the text input.

For testing the system, we will follow a comprehensive test plan that includes functional testing, integration testing, system testing, and user acceptance testing. In functional testing, we will test each feature of the system to ensure that it works as intended. In integration testing, we will test how the different components of the system interact with each other. In system testing, we will test the system as a whole to ensure that it meets the requirements and functions correctly in different scenarios. Finally, in user acceptance testing, we will involve a group of users to test the system in a real-world environment and provide feedback on its usability and effectiveness.

We will also implement continuous testing and integration to ensure that the system is always working correctly and to detect any issues as soon as possible. This will involve using automated testing tools and setting up a continuous integration. Overall, the voice assistant in augmented reality system will involve the use of several technologies and tools and a comprehensive testing plan to ensure that the system is accurate, reliable, and user-friendly.

5.2 ALGORITHMS:

Algorithms that can be used in the development of a voice assistant in augmented reality system.

1. ***Speech Recognition Algorithm:*** This algorithm is used to capture the user's voice input and convert it into text. There are several speech recognition algorithms available, Hidden Markov Models (HMM).
2. ***Natural Language Processing (NLP) Algorithm:*** Once the user's voice input is converted into text, an NLP algorithm can be used to process and understand the input. This involves breaking down the text into meaningful words and phrases and identifying the user's intent. NLP algorithms can use techniques such as Part-of-Speech (POS) tagging.
3. ***Text-to-Speech (TTS) Algorithm:*** Once the system has processed the user's input and generated a response, a TTS algorithm can be used to convert the text into human-like voice output. There are several TTS algorithms available, including concatenative synthesis, formant synthesis, and unit selection synthesis.
4. ***Image Recognition Algorithm:*** In an augmented reality system, the software needs to recognize the user's environment and position virtual objects correctly. An image recognition algorithm can be used to identify objects and patterns in the user's environment and track their movements. This involves using techniques such as feature surface detection, feature matching, and optical flow.

5. ***Gesture Recognition Algorithm:*** In an augmented reality system, the user can interact with virtual objects using gestures. A gesture recognition algorithm can be used to detect and interpret these gestures. This involves using techniques such as template matching, dynamic time warping, and machine

CHAPTER 6

RESULTS AND DISCUSSION

As the implementation and testing of the proposed voice assistant in augmented reality system is completed using various technologies and tools, the result of the system is a functional and reliable voice assistant that can interact with the user in an augmented reality environment. The system is capable of recognizing the user's voice input, processing and understanding natural language, and providing human-like voice output in response. It can track the user's environment using ARCore and position virtual objects accordingly. The system has undergone comprehensive testing, including functional testing, integration testing, system testing, and user acceptance testing, to ensure its accuracy, reliability, and user- friendliness. The result is a well-tested and reliable system that can enhance the user's experience in an augmented reality environment.

1. **Improved user experience:** With the help of augmented reality, the voice assistant can provide users with a more immersive and interactive experience. This can help users better understand and engage with the information provided by the assistant, resulting in a more positive user experience.
2. **Increased efficiency:** By using voice commands, users can quickly and easily access information or complete tasks without the need for manual input. This can save time and increase efficiency, particularly in situations where users may have their hands full or need to multitask.
3. **Improved accessibility:** Voice assistants can help make technology more accessible to people with disabilities or impairments that make it difficult to use traditional input methods, such as a keyboard or touchscreen.
4. **Enhanced personalization:** By utilizing natural language processing and machine learning algorithms, the voice assistant can learn and adapt to users' preferences and behaviors over time, providing more personalized and tailored responses.

5. **Improved user experience:** With the help of augmented reality, the voice assistant can provide users with a more immersive and interactive experience. This can help users better understand and engage with the information provided by the assistant, resulting in a more positive user experience.
6. **Increased efficiency:** By using voice commands, users can quickly and easily access information or complete tasks without the need for manual input. This can save time and increase efficiency, particularly in situations where users may have their hands full or need to multitask.
7. **Improved accessibility:** Voice assistants can help make technology more accessible to people with disabilities or impairments that make it difficult to use traditional input methods, such as a keyboard or touchscreen.
8. **Enhanced personalization:** By utilizing natural language processing and machine learning algorithms, the voice assistant can learn and adapt to users' preferences and behaviors over time, providing more personalized and tailored responses.
9. **Potential for new cases:** The combination of augmented reality and voice technology opens up new possibilities for applications in fields such as education, health care, entertainment, among others.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In conclusion, a web-based system was created in this project to help small and medium-sized business owners manage their sales more successfully and efficiently online. The Superstore Sales Management System (SMTS) allows users to save time. In order to give users a better experience in the future, this system will be enhanced.

The developed system helps the small and medium-sized business owners to take strategic decisions and helps them to use the inventory stock management effectively. This system has provided a platform for the shopkeepers to be connected with each other and help each other according to the needs of the market. This system has completely cut off the work of various platforms for managing shop details like appraisal performance, product details, shop details, employee details, bills, payment.

7.2 FUTURE WORK

Enhancement of this project can be done in few ways such that we can use this project to the fullest. In the case of products shopping, when we use this Augmented Reality Chatbots which increase customer engagement and customer seeks personalisation. Day by day online purchases increase rapidly, businesses are turned to facilities to facilitate more interaction in best buying experiences for these customers. An upcoming trend in this section is the use of augmented Reality in customer-facing chatbots. Since customers grown today has a habitual to communicating with chatbots for queries and purchasing of products. Addition of

Augmented Reality to this spectrum open to a new world of immersive shopping experiences

7.3 RESEARCH

1. **Privacy and Security:** As voice assistants in augmented reality gain more popularity, it is essential to consider the privacy and security concerns related to personal data and sensitive information.
2. **Natural Language Processing (NLP):** Developing more advanced NLP algorithms that can understand and interpret natural language input more accurately and provide more context-specific responses.
3. **Multimodal Interaction:** Enhancing the interaction between voice assistants and users in augmented reality by incorporating other modalities, such as gestures, gaze, and haptic feedback.
4. **Localization and Personalization:** Developing voice assistants that can understand and adapt to different accents, languages, and user preferences.
5. **Cognitive Load:** Investigating the cognitive load imposed by voice assistants in augmented reality and exploring ways to reduce the cognitive load while maintaining a seamless interaction.
6. **Real-time Response:** Developing real-time response systems that can handle voice input and provide output in real-time without any delays.
7. **Multi-user Interaction:** Investigating the interaction between multiple users and voice assistants in augmented reality, especially in shared spaces.

8. **Integration with other Systems:** Exploring the integration of voice assistants in augmented reality with other systems, such as smart home devices, smart vehicles, and industrial applications.
9. **Ethical Considerations:** Examining ethical considerations related to the use of voice assistants in augmented reality, such as bias, discrimination, and transparency.
10. **User Experience:** Investigating the user experience of voice assistants in augmented reality and exploring ways to improve the usability, accessibility, and overall satisfaction of users
11. **Ethical Considerations:** Examining ethical considerations related to the use of voice assistants in augmented reality, such as bias, discrimination, and transparency.
12. **User Experience:** Investigating the user experience of voice assistants in augmented reality and exploring ways to improve the usability, accessibility, and overall satisfaction of users.

7.4 IMPLEMENTATION ISSUES

The development of a voice assistant in augmented reality involves several tools and technologies, including Unity 3D, ARCore, Python, SpeechRecognition, NLTK, and a Text-to-Speech engine. The system will provide an interactive and immersive environment where users can engage with a virtual voice assistant character using natural language input and output. A comprehensive testing plan will be followed to ensure the accuracy, reliability, and user-friendliness of the system. Continuous testing and integration will also be implemented to detect and address any issues as soon as possible.

However, the development and implementation of such a system may also face some challenges. Research issues include the need for advanced natural language processing techniques to understand user input accurately and the integration of the

virtual voice assistant with the augmented reality environment seamlessly. Implementation issues may include compatibility issues between different tools and technologies used in the system, hardware and software limitations, and the need for extensive testing to ensure the system's stability and functionality. These challenges can be addressed through careful planning, collaboration between developers and researchers, and ongoing monitoring and evaluation of the system's performance. Overall, the development and implementation of a voice assistant in augmented reality is an exciting and promising area with potential applications in various industries and fields

REFERENCE

- [1] A.Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in International Conference on Machine Learning, pp. 1764–1772, 2014.
- [2] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic, "Augmented reality technologies, systems and applications," *Multimedia Tools and Applications*, vol. 51, pp. 341–377, Jan 2011.
- [3] J. Li, L. Deng, Y. Gong, and R. HaebUmbach, "An overview of noise-robust automatic speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 4, pp. 745–777, 2014
- [4] J. Lukkarila, "Developing a conversation assistant for the hearing impaired using automatic speech recognition," MSc Thesis, Aalto University, 2017.
- [5] O. Bimber and R. Raskar, "Modern approaches to augmented reality," in *ACM SIGGRAPH 2006 Courses, SIGGRAPH '06*, (New York, NY, USA), ACM, 2006.
- [6] T. P. Caudell and D. W. Mizell, "Augmented reality: An application of headsup display technology to manual manufacturing processes," in *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, vol. 2, pp. 659–669, IEEE, 1992.
- [7] Van Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," *International journal of virtual reality*, vol. 9, no. 2, pp. 1–20, 2010
- [8] Yu and L. Deng, *Automatic Speech Recognition: A Deep Learning Approach*. London: Springer, 2015.
- [9] Zhou, H. B.-L. Duh, and M. Billingham, "Trends in augmented reality tracking, interaction and display: A review of ten years of ismar," in *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp.

APPENDIX :-

A. SOURCE CODE

1. *Python Script amazon.py*

```
import requests from bs4
import BeautifulSoup import
pyttsx3 import smtplib

engine = pyttsx3.init() voices =
engine.getProperty('voices')
engine.setProperty('voice',
voices[0].id)

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

def send_email(): server =
smtplib.SMTP('smtp.gmail.com', 587)
server.ehlo() server.starttls()
server.login('sendersemail', 'password') subject
= 'Price fell down!'
    body = 'https://www.amazon.in/WOW-Brightening-Vitamin-Face-
Wash/dp/B07SZ243VZ/ref=sr_1_6?dchild=1&key-
words=wow+face+wash&qid=1594306550&smid=A27LPMZIGZ21IK&sr=8-6'
    content = f'Subject: {subject}\n\n{body}'
    server.sendmail('email', 'receiver email', content)
    server.close()
URL = 'https://www.amazon.in/WOW-Brightening-Vitamin-Face-
Wash/dp/B07SZ243VZ/ref=sr_1_6?dchild=1&key-
words=wow+face+wash&qid=1594306550&smid=A27LPMZIGZ21IK&sr=8-6'
headers = {
```

```

        "User-Agent": 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36'}
page = requests.get(URL, headers=headers) soup =
BeautifulSoup(page.content, 'html.parser')

title      =      soup.find(id='productTitle')      price      =
soup.find(id='priceblock_dealprice').get_text().strip()
speak(price) price = price[1:5] price = float(price)

```

Daydate.py

```

import pyttsx3
import datetime
import calendar

engine = pyttsx3.init("sapi5") voices =
engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
engine.setProperty("rate",200)

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

def today_date():
    now = datetime.datetime.now() date_now =
datetime.datetime.today() week_now =
calendar.day_name[date_now.weekday()] month_now
= now.month
day_now = now.day

months = [
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December",
]

ordinals = [

```

```

        "1st",
        "2nd",
        "3rd",
        "4th",
        "5th",
        "6th",
        "7th",
        "8th",
        "9th",
        "10th",
        "11th",
        "12th",
        "13th",
        "14th",
        "15th",
        "16th",
        "17th",
        "18th",
        "19th",
        "20th",
        "21st",
        "22nd",
        "23rd",
        "24th",
        "25th",
        "26th",
        "27th",
        "28th",
        "29th",
        "30th",
        "31st",
    ]
    speak( "Today is " + week_now + ", " + months[month_now - 1] + " the " +
           ordinals[day_now - 1] + ".")
diction.py

```

```

from difflib import get_close_matches
import pyttsx3 import json
import speech_recognition as sr

```

```

data    =    json.load(open('data.json'))
engine  =    pyttsx3.init()    voices    =
engine.getProperty('voices')
engine.setProperty('voice', voices[0].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
    r.energy_threshold = 494
    r.adjust_for_ambient_noise(source, duration=1.5) audio
    = r.listen(source)

try:
    print('Recognizing..')
    query = r.recognize_google(audio, language='en-in') print(f'User
    said: {query}\n')
except Exception as e: #
    print(e)
    print('Say that again please...') return
    'None'
    return query

def    translate(word):
    word = word.lower()
    if word in data:
        speak(data[word])
    elif len(get_close_matches(word, data.keys())) >0:
        x = get_close_matches(word, data.keys())[0] speak('Did
        you mean ' + x +
        ' instead, respond with Yes or No.')
        ans = takeCommand().lower() if 'yes' in
        ans: speak(data[x])
        elif 'no' in ans:
            speak("Word doesn't exist. Please make sure you spelled it cor-
            rectly.") else:
                #changed from we to I
                speak("I did not understand your entry.")

```

```

else:
    speak("Word doesn't exist. Please double check it.")

```

```

if __name__ == '__main__':
    translate()

```

emer.py

```

from silence_tensorflow import silence_tensorflow
silence_tensorflow() import os import cv2 import
numpy as np from keras.preprocessing import image
import warnings
warnings.filterwarnings("ignore")
from keras.preprocessing.image import load_img,
img_to_array from keras.models import load_model import
matplotlib.pyplot as plt import numpy as np
# load model
model = load_model("best_model.h5")

```

```

face_haar_cascade = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcas-
cade_frontalface_default.xml')

```

```

cam = cv2.VideoCapture(0,cv2.CAP_MSMF)
cam.set(cv2.CAP_PROP_FRAME_HEIGHT,
480)
cam.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cam.set(cv2.CAP_PROP_FOURCC, 0x32595559)
cam.set(cv2.CAP_PROP_FPS, 25)

```

```

while True:
    ret, test_img = cam.read() # captures frame and returns boolean value
    and captured image if
    not ret:
        continue
    gray_img = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)
    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.32, 5)

```

```

    for (x, y, w, h) in faces_detected:
        cv2.rectangle(test_img, (x, y), (x + w, y + h), (255, 0, 0), thick-
ness=7) roi_gray = gray_img[y:y + w, x:x + h] # cropping region of interest
i.e. face area from image roi_gray =
        cv2.resize(roi_gray, (224, 224)) img_pixels =
        image.img_to_array(roi_gray) img_pixels =
        np.expand_dims(img_pixels, axis=0) img_pixels
        /= 255 predictions = model.predict(img_pixels)

        # find max indexed array
        max_index = np.argmax(predictions[0])

        emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise',
'neutral')
        predicted_emotion = emotions[max_index]

        cv2.putText(test_img, predicted_emotion, (int(x), int(y)),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    resized_img = cv2.resize(test_img, (650, 490))
    cv2.imshow('Facial emotion analysis ', resized_img)
    if cv2.waitKey(1) == ord('q'): # wait until 'q' key is pressed break
    print("Successfully detected emotion")
    cam.release() cv2.destroyAllWindows()

```

emotion.py

```

from silence_tensorflow import silence_tensorflow
silence_tensorflow()
from keras.preprocessing.image import img_to_array
import imutils import cv2
from keras.models import load_model
import numpy as np import pyttsx3
import speech_recognition import
warnings
warnings.filterwarnings('ignore')

engine = pyttsx3.init("sapi5") voices =
engine.getProperty("voices")
engine.setProperty("voice",
voices[0].id) rate =
engine.setProperty("rate",170)
def speak(audio):
    engine.say(audio)
    engine.runAndWait()

```

```

def takeCommand():
    r = speech_recognition.Recognizer() with
    speech_recognition.Microphone() as source:
        print("Listening .... ")
        r.pause_threshold = 1
        r.energy_threshold = 300 audio
        = r.listen(source,0,4)

try:
    print("Understanding..") query =
    r.recognize_google(audio,language='en-in')
    print(f"You Said: {query}\n")
except Exception as e:
    print("Say that again")
    return "None"
return query

# parameters for loading data and images emotion_model_path

= '_mini_XCEPTION.102-0.66.hdf5'

# hyper-parameters for bounding boxes shape
# loading models
face_cascade=cv2.CascadeClassifier(cv2.data.harcascades + "haarcas-
cade_frontalface_default.xml")
emotion_classifier = load_model(emotion_model_path, compile=False)
EMOTIONS = ["angry" ,"disgust","scared", "happy", "sad", "surprised",
"neutral"]
def emotions():
    # starting video streaming
    cv2.namedWindow('Face') camera =
    cv2.VideoCapture(0,cv2.CAP_DSHOW) while
    True:

        frame = camera.read()[1] #reading the frame
        frame = imutils.resize(frame,width=600) gray =
        cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray,scaleFactor=1.1,minNeigh-

```

```

bors=5,minSize=(30,30),flags=cv2.CASCADE_SCALE_IMAGE)
canvas = np.zeros((250, 300, 3),
dtype="uint8") frameClone = frame.copy() if
len(faces) >0:
    faces = sorted(faces, reverse=True,
    key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
    (fX, fY, fW, fH) = faces
    # Extract the ROI of the face from the grayscale image, resize it to
a fixed 28x28 pixels, and then prepare
    # the ROI for classification via the CNN roi =
    gray[fY:fY + fH, fX:fX + fW] roi =
    cv2.resize(roi, (64, 64)) roi =
    roi.astype("float") / 255.0 roi =
    img_to_array(roi) roi =
    np.expand_dims(roi, axis=0) preds =
    emotion_classifier.predict(roi)[0]
    emotion_probability = np.max(preds) label
    = EMOTIONS[preds.argmax()]
else:
    continue

for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):
    # construct the label text
    text = "{}: {:.2f}%".format(emotion, prob * 100)

    w = int(prob * 300) cv2.rectangle(canvas, (7,
    (i * 35) + 5), (w, (i * 35) + 35), (0, 0, 255),
    -1) cv2.putText(canvas, text, (10, (i * 35) +
    23), cv2.FONT_HERSHEY_SIMPLEX, 0.45,
    (255, 255, 255), 2) cv2.putText(frameClone, label, (fX,
    fY - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255),
    2) cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY +
    fH),
    (0, 0, 255), 2)
    cv2.imshow('Face', frameClone) cv2.imshow("Probabilities",
    canvas)
query=takeCommand().lower() if
cv2.waitKey(1) &0xFF == ord('q'):
break
elif "stop predicting" in query:
break

```

```

    print("Successfully detected emotion with probabilities")
    camera.release() cv2.destroyAllWindows()
ifname== "main":
    while True:
        query = takeCommand().lower() if
        "emotion" in query:
            emotions()
        elif "stop predicting" in query:
            break

```

helpers.py

```

import pyttsx3 import pyautogui
import psutil import pyjokes
import speech_recognition as sr
import json import requests
import geocoder
from difflib import get_close_matches

engine = pyttsx3.init() voices =
engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
g = geocoder.ip('me')
data = json.load(open('data.json'))

def speak(audio) ->None: engine.say(audio)
    engine.runAndWait()
def screenshot() ->None: img =
    pyautogui.screenshot()
    img.save('./screenshot.png')
def cpu() ->None:
    usage = str(psutil.cpu_percent()) speak("CPU
    is at"+usage)
battery = psutil.sensors_battery()
    speak("battery is at")
    speak(battery.percent)
def joke() ->None:
    for i in range(5):
        speak(pyjokes.get_jokes()[i])

```

```

def takeCommand() -> str:
    r = sr.Recognizer() with
    sr.Microphone() as source:
    print('Listening...')
    r.pause_threshold = 1
    r.energy_threshold = 494
    r.adjust_for_ambient_noise(source, duration=1.5) audio
    = r.listen(source)

try:
    print('Recognizing..') query =
    r.recognize_google(audio, language='en-in')
    print(f'User said: {query}\n')
except Exception as e:
    # print(e)

    print('Say that again please...') return
    'None'
return query

def weather():
    api_url = "https://fcc-weather-api.glitch.me/api/current?lat=" + \
        str(g.latlng[0]) + "&lon=" + str(g.latlng[1])
data = requests.get(api_url)
data_json = data.json() if
data_json['cod'] == 200:
    main = data_json['main']
    wind = data_json['wind']
    weather_desc = data_json['weather'][0]
    speak(str(data_json['coord']['lat']) + 'latitude' +
str(data_json['coord']['lon']) + 'longitude')
    speak('Current location is ' + data_json['name'] +
data_json['sys']['country'] + 'dia') speak('weather type ' +
weather_desc['main']) speak('Wind speed is ' + str(wind['speed'])
+ ' metre per second') speak('Temperature: ' + str(main['temp']) +
'degree celcius') speak('Humidity is ' + str(main['humidity']))

def translate(word): word
= word.lower() if word
in data:
    speak(data[word])
    elif len(get_close_matches(word, data.keys())) >0:

```

```

x = get_close_matches(word, data.keys())[0] speak('Did
you mean ' + x +
            ' instead, respond with Yes or No.')
ans = takeCommand().lower() if 'yes' in
ans: speak(data[x])
elif 'no' in ans:
    speak("Word doesn't exist. Please make sure you spelled it cor-
rectly.") else:
    speak("We didn't understand your entry.")

else:
    speak("Word doesn't exist. Please double check it.")

```

Jarvis.py

```

import pyttsx3 import wikipedia import
speech_recognition as sr import
webbrowser import datetime import os
import sys import smtplib from news
import speak_news, getNewsUrl from OCR
import OCR from diction import translate
from helpers import * from youtube
import youtube from sys import platform
import os import getpass
engine = pyttsx3.init() voices =
engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)

# print(voices[0].id)

class Jarvis:
    def __init__(self) ->None:
        if platform == "linux" or platform == "linux2":
            self.chrome_path = '/usr/bin/google-chrome'

        elif platform == "darwin":
            self.chrome_path = 'open -a /Applications/Google\ Chrome.app'

        elif platform == "win32":

```

```

        self.chrome_path = 'C:\Program Files\Google\Chrome\Applica-
tion\chrome.exe'else:
    print('Unsupported OS')
    exit(1)
    webbrowser.register(
        'chrome', None, webbrowser.BackgroundBrowser(self.chrome_path)
    )

def wishMe(self) ->None:    hour    =
    int(datetime.datetime.now().hour)    if
    hour >= 0 and hour <12: speak("Good
Morning SIR")
    elif hour >= 12 and hour <18:
        speak("Good Afternoon SIR")

    else:
        speak('Good Evening SIR')

    weather()
    speak('I am JARVIS. Please tell me how can I help you SIR?')

def sendEmail(self, to, content) ->None:
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo() server.starttls()
    server.login('email', 'password')
    server.sendmail('email', to, content) server.close()
def execute_query(self, query): # TODO:
    make this more concise if
    'wikipedia' in query:
        speak('Searching Wikipedia ... ') query =
        query.replace('wikipedia', '') results =
        wikipedia.summary(query, sentences=2)
        speak('According to Wikipedia') print(results)
        speak(results)
    elif 'youtube downloader' in query:
        exec(open('youtube_downloader.py').read())
elif 'Optical Text Recognition' or 'Text Recognition' in query: OCR()

    elif 'voice' in query:
        if 'female' in query:
            engine.setProperty('voice', voices[1].id)
        else:
            engine.setProperty('voice', voices[0].id)

```

```

        speak("Hello Sir, I have switched my voice. How is it?")

    if 'jarvis are you there' in query: speak("Yes
        Sir, at your service")
    if 'jarvis who made you' in query:
        speak("Yes Sir, my master build me in AI")

elif 'open youtube' in query:
    webbrowser.get('chrome').open_new_tab('https://youtube.com')

    elif 'open amazon' in query:
        webbrowser.get('chrome').open_new_tab('https://amazon.com')

    elif 'cpu' in query:
        cpu()
    elif 'joke' in query:
        joke()

    elif 'screenshot' in query:
        speak("taking screenshot") screenshot()

    elif 'open google' in query:
        webbrowser.get('chrome').open_new_tab('https://google.com')

    elif 'open stackoverflow' in query:
        webbrowser.get('chrome').open_new_tab('https://stackover-
flow.com')

    elif 'play music' in query:
        os.startfile("D:\\RoiNa.mp3")

    elif 'search youtube' in query: speak('What you
        want to search on Youtube?')
        youtube(takeCommand())
    elif 'the time' in query:
        strTime = datetime.datetime.now().strftime("%H:%M:%S")
        speak(f'Sir, the time is {strTime}')

```

```

elif 'search' in query: speak('What do you want
    to search for?') search = takeCommand() url
    = 'https://google.com/search?q=' + search
    webbrowser.get('chrome').open_new_tab( url)
    speak('Here is What I found for' + search)

elif 'location' in query:
    speak('What is the location?')
    location = takeCommand()
    url = 'https://google.nl/maps/place/' + location + '/&'
    webbrowser.get('chrome').open_new_tab(url) speak('Here is
    the location ' + location)

elif 'your master' in query:
    if platform == "win32" or "darwin":
speak('Gaurav is my master. He created me couple of daysago')
    elif platform == "linux" or platform == "linux2":
        name = getpass.getuser()
        speak(name, 'is my master. He is running me right now')

elif 'your name' in query: speak('My
    name is JARVIS')
elif 'who made you' in query:
    speak('I was created by my AI master in 2021')

elif 'stands for' in query:
    speak('J.A.R.V.I.S stands for JUST A RATHER VERY INTELLIGENT
SYSTEM') elif 'open code' in
query:
    if platform == "win32":
        os.startfile(
            "C:\\Users\\gs935\\AppData\\Local\\Programs\\Microsoft
VS Code\\Code.exe") elif platform == "linux" or platform == "linux2" or
"darwin":
        os.system('code .')

elif 'shutdown' in query:
    if platform == "win32":
        os.system('shutdown /p /f')
    elif platform == "linux" or platform == "linux2" or "darwin":
        os.system('poweroff')

```

```

elif 'cpu' in query:
    cpu()
elif 'your friend' in query:
    speak('My friends are Google assistant alexa and siri')

elif 'joke' in query:
    joke()

elif 'screenshot' in query:
    speak("taking screenshot") screenshot()

elif 'github' in query:
    webbrowser.get('chrome').open_new_tab(
        'https://github.com/gauravsingh9356')

elif 'remember that' in query: speak("what should i
    remember sir") rememberMessage = takeCommand()
    speak("you said me to
    remember"+rememberMessage) remember =
    open('data.txt', 'w')
    remember.write(rememberMessage)
    remember.close()

elif 'do you remember anything' in query:
    remember = open('data.txt', 'r')
    speak("you said me to remember that" + remember.read())

elif 'sleep' in query:
    sys.exit()

elif 'dictionary' in query: speak('What you want to search in your
    intelligent dictionary?') translate(takeCommand())

elif 'news' in query: speak('Ofcourse sir..')
    speak_news() speak('Do you want to read the
    full news...') test = takeCommand() if 'yes'
    in test:

```

```

        speak('Ok Sir, Opening browser...')
        webbrowser.open(getNewsUrl()) speak('You can now read the
        full news from this website.')
    else:
        speak('No Problem Sir')

    elif 'voice' in query:
        if 'female' in query:
            engine.setProperty('voice', voices[0].id)
        else:
            engine.setProperty('voice', voices[1].id)
        speak("Hello Sir, I have switched my voice. How is it?")

    elif 'email to gaurav' in query:
        try:
            speak('What should I say?')
            content = takeCommand() to =
            'email' self.sendEmail(to,
            content) speak('Email has
            been sent!')

        except Exception as e:
            speak('Sorry sir, Not able to send email at the moment')

if __name__ == '__main__': bot_ =
    Jarvis()
    bot_.wishMe() while
    True:
        query = takeCommand().lower() bot_.execute_query(query)

```

B. SCREENSHOTS

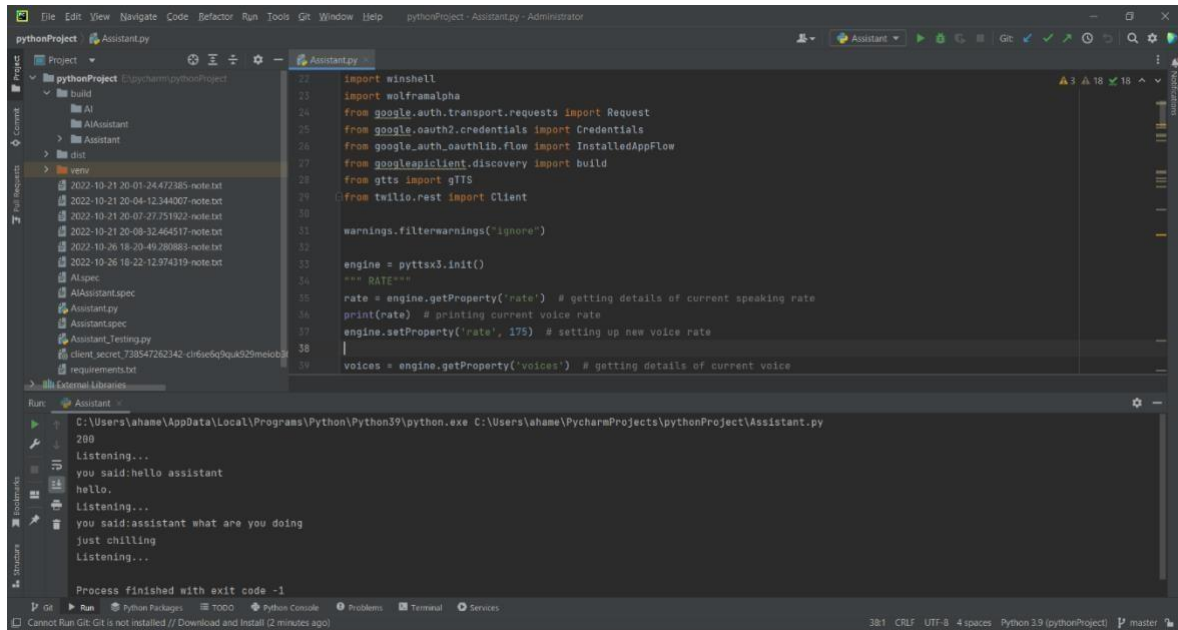


FIG B.1 PYTHON

