SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-VXXXX
Volume X Issue X, 1-4, Month 2023

Type of Article *(Original Article)*

Smart Desktop Assistant based on Speech Recognition Technique using AI

Adnaan Zubair Shah[1], Abdul Mahatab Sarban[2], Rohit Dinesh Shelar[3], R. A. Khedkar[4]
[1]*Electronics and Telecommunication Engineering department, Sinhgad Academy Of Engineering Pune, India*
[2]*Electronics and Telecommunication Engineering department, Sinhgad Academy Of Engineering Pune, India*
[3]*Electronics and Telecommunication Engineering department, Sinhgad Academy Of Engineering Pune, India*
[4]*Electronics and Telecommunication Engineering department, Sinhgad Academy Of Engineering Pune, India*

[1] adnaanzshah@gmail.com [2] sarbanabdul@gmail.com [3] rohitshelar99332@gmail.com [4] rakhi08.khedkar@gmail.com

Received: Revised: Accepted: Published:

Abstract - Now a days due to globalization swift communication and quick outcome based tasks are very important. In such environment physically/visually disabled persons needs a personal assistant that can help them to perform various online and offline operations and various tasks quickly as per requirement. Focusing on this in this investigation Smart Desktop Assistant based on Voice and Speech Recognition Technique using AI is proposed. This system can significantly transform human-computer interaction. This assistant offer hands-free access to services, benefiting physically disabled users and enhancing productivity across various fields. The physically/visually disabled persons can open and handle various tool installed on desktop. With advancements in artificial intelligence and natural language processing, personalized and context-aware assistance is now possible, revolutionizing human-computer interaction.

Keywords **- Smart Desktop Assistant; Speech Recognition; Artificial Intelligence; Natural Language Processing; Voice-Controlled Applications; Python Programming; Task Automation; Human-Computer Interaction; Assistive Technology; JARVIS Project**

1.Introduction

Voice assistants have become ubiquitous in modern society, revolutionizing the way we interact with technology. These assistants, powered by artificial intelligence and natural language processing, offer convenience and efficiency in various tasks, ranging from setting reminders to controlling smart home devices. However, existing voice assistants often lack flexibility and customization options for users. The JARVIS Voice Assistant project aims to address these limitations by providing an open-source voice recognition system that can be easily modified and enhanced by users according to their specific needs and preferences. Times New Roman

The motivation behind developing the JARVIS Voice Assistant stems from the growing demand for personalized and adaptable virtual assistants. By leveraging the capabilities of Python and various modules, the project seeks to empower users to automate routine tasks and streamline their workflow. Unlike proprietary voice assistants that may require costly subscriptions or limited customization options, JARVIS offers a cost-effective and customizable solution for individuals and businesses alike.

Through this research paper, we aim to explore the development process, methodologies, and outcomes of the JARVIS Voice Assistant project. By providing a comprehensive overview of the project's objectives and significance, we seek to highlight its potential impact on society and pave the way for future advancements in voice assistant technology.

2. Related Work

Prior research in the field of voice assistants has laid the foundation for projects like JARVIS. Studies have investigated various aspects of voice recognition, natural language understanding, and user interaction to enhance the capabilities of virtual assistants. For example, research on Raspberry Pi-based personal voice assistants has demonstrated the feasibility of using affordable hardware for voice recognition and text-to-speech conversion. Similarly, studies on home automation systems have explored the integration of IoT, NLP, and machine learning to create versatile and user-friendly smart home solutions. While existing research provides valuable insights into the development and functionality of voice assistants, there is still room for innovation and improvement. The JARVIS Voice Assistant project seeks to build upon the achievements of previous research by offering a customizable and adaptable voice recognition system tailored to desktop environments. By examining the strengths and limitations of related work, we can identify opportunities to enhance the

performance, usability, and accessibility of the JARVIS Voice Assistant.
By conducting a thorough review of related work, we can gain a deeper understanding of the challenges and opportunities in the field of voice assistant technology. By building upon the foundations laid by previous research, the JARVIS Voice Assistant project aims to push the boundaries of what is possible in terms of voice recognition, natural language processing, and user interaction.

In [1], it was proposed that a personal assistant system utilized speech recognition algorithms to enable voice-controlled interactions. In [2], the study discussed the development of a virtual personal assistant specifically designed to assist visually impaired individuals, enhancing accessibility and independence. In [3], the paper presented an AI-powered smart assistant aimed at enhancing user productivity through voice recognition and natural language processing technologies. In [4], the research explored the implementation of voice recognition technology in Python programming, facilitating efficient and intuitive coding practices. In [5], the study introduced an innovative assistive system leveraging Raspberry Pi technology to aid visually impaired individuals in accessing information and performing daily tasks.
In [6], the paper explored the evolution of virtual personal assistants such as Microsoft Cortana, Apple Siri, Amazon Alexa, and Google Home, highlighting their features, applications, and future developments. In [7], the paper presented a personal assistant system incorporating voice recognition intelligence through a speech recognition algorithm. In [8], the study introduced a virtual personal assistant designed specifically for visually impaired individuals to enhance accessibility and independence. In [9], the paper presented an AI-powered smart assistant aimed at enhancing user productivity through voice recognition and natural language processing technologies. In [10], the research explored the implementation of voice recognition technology in Python programming to facilitate efficient and intuitive coding practices.

In the previous research the systems were not developed for physically handicapped persons. Hence, in the proposed research a novel system is developed which can provide easy access to desktop by physically handicap persons. The proposed system is explained next section

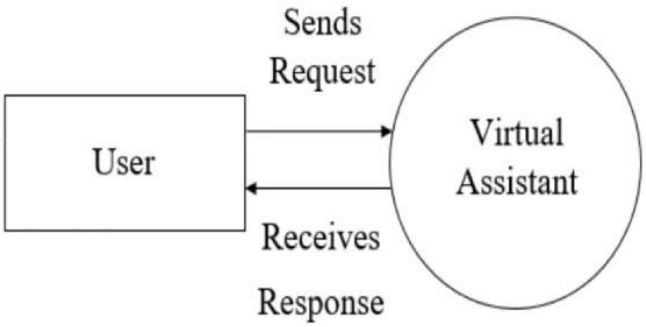
3. Proposed Work

AI desktop voice assistant system: The proposed work encompasses the development and implementation of the **JARVIS** Voice Assistant, a customizable and open-source voice recognition system. This project aims to provide users with an intuitive and efficient way to interact with their desktop environment using voice commands. Unlike traditional voice assistants that may have limited functionality or require internet connectivity, JARVIS operates as a standalone application, offering users greater control and privacy over their data.

Objectives:
1. To enhance various functionalities with standard libraries for desktop assistant.
2. To integrate various AI Algorithm (NLP Algorithms) to performs various tasks.
3. To expand Desktop Assistant capabilities with the help of external APIs.
4. To establish interactive voice interaction.
5. To improve productivity and efficiency of Desktop Assistant.

The core objectives of the proposed work include designing a user-friendly interface for voice interaction, implementing robust speech recognition algorithms, and integrating various modules for task automation and information retrieval. By leveraging the power of Python and its libraries, the JARVIS Voice Assistant project seeks to deliver a seamless and responsive user experience. Additionally, the project prioritizes customization and extensibility, allowing users to tailor the assistant to their specific needs and preferences.
Key features of the proposed work include voice-controlled application launching, web browsing, media playback, and system utilities. Through a combination of text-to-speech and speech-to-text conversion, JARVIS enables users to perform a wide range of tasks hands-free, thereby increasing productivity and accessibility. Furthermore, the project emphasizes accessibility by providing support for users with visual or motor impairments, ensuring that everyone can benefit from the capabilities of the voice assistant.

4. Methadology



The methodology employed in developing the JARVIS Voice Assistant project involves a multi-stage process that encompasses requirement analysis, design, implementation, testing, and deployment. Initially, the project team conducted extensive research to identify user needs and preferences regarding voice assistant functionality. This phase involved studying existing voice assistant systems, analyzing user feedback, and defining the scope and objectives of the project.
Once the requirements were established, the design phase began, wherein the project team outlined the architecture, user interface, and key features of the voice assistant.
This phase also involved selecting suitable technologies and

frameworks for implementing the various components of the system. Python was chosen as the primary programming language due to its versatility, ease of use, and extensive library support.
The implementation phase involved writing code to realize the design specifications outlined in the previous phase. This included developing modules for speech recognition, natural language understanding, task execution, and user interaction. Throughout the implementation process, the project team followed best practices in software engineering, including modular design, code documentation, and version control.

5. Algorithms

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FOR ASSISTIVE TECHNOLOGY

1) Module: Speak (Text-to-Speech Conversion)

Algorithm Steps:

1. **Import Libraries**: Use pyttsx3 for offline text-to-speech conversion.
2. **Initialize Engine**: Initialize the text-to-speech engine with pyttsx3.init().
3. **Set Voice**: Choose a preferred voice using engine.setProperty('voice', voices[0].id) for a male voice.
4. **Define Speak Function**: Create a function to convert text to speech.
5. **Call Speak Function**: Use the function to speak the desired text.

AI and ML Relevance:

- **Artificial Intelligence**: The use of text-to-speech conversion is a fundamental AI capability, enhancing human-computer interaction.
- **Helps Blind and Physically Disabled People**: Converts written text to spoken words, enabling visually impaired individuals to access written information.

Code:

```
import pyttsx3

def speak(text):
    engine = pyttsx3.init()
    voices = engine.getProperty('voices')
    engine.setProperty('voice', voices[0].id)
    engine.say(text)
    engine.runAndWait()
```

2) Module: Wish me

Algorithm Steps:

1. **Get Current Hour**: Retrieve the current hour using datetime.datetime.now().hour.
2. **Conditional Greeting**: Greet the user based on the time of day (morning, afternoon, evening).
3. **Speak Greeting**: Use the text-to-speech engine to deliver the greeting.

AI and ML Relevance:

- **Artificial Intelligence**: Time-based contextual responses improve user interaction.
- **Helps Blind and Physically Disabled People**: Provides timely greetings and assists in maintaining a routine.

Code:

```
import datetime

def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour >= 0 and hour < 12:
        speak("Good Morning!")
    elif hour >= 12 and hour < 18:
        speak("Good Afternoon!")
    else:
        speak("Good Evening!")
    speak("I am Jarvis. How can I assist you today?")
```

3) Module: Take Command (Speech Recognition)

Algorithm Steps:

1. **Import Libraries**: Use speech_recognition for speech-to-text conversion.
2. **Capture Audio**: Listen to the microphone input and capture audio.
3. **Recognize Speech**: Convert audio to text using Google Speech Recognition.
4. **Return Command**: Return the recognized text as the command.

AI and ML Relevance:

- **Machine Learning**: Utilizes advanced ML models for accurate speech recognition.
- **Helps Blind and Physically Disabled People**: Enables hands-free control of devices through voice commands.

Code:

```
import speech_recognition as sr

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"User said: {query}\n")
    except Exception as e:
        print("Say that again please...")
        return "None"
    return query
```

3) Module: Send Email

This Module hasn't been performed in the program

Algorithm Steps:
1.Import the smtplib library for sending emails.
2.Define the sendEmail() function with parameters for the recipient's email address (to) and email content (content).
3.Establish a connection with the SMTP server of the email service provider (e.g., Gmail) using smtplib.SMTP().
4.Set up the SMTP connection using starttls() method for secure transmission.
5.Login to the sender's email account using login() with the sender's email address and password.
6.Send the email using sendmail() with the sender's email address, recipient's email address, and email content.
7.Close the SMTP connection using close().

Code:
```
import smtplib

def sendEmail(to, content):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login('your_email@gmail.com', 'your_password')
    server.sendmail('your_email@gmail.com', to, content)
    server.close()
```

4) Module: PDF Reader

Algorithm Steps:

1. **Import Libraries**: Use PyPDF2 for reading PDF files.
2. **Define Reader Function**: Create a function to read and extract text from PDF files.
3. **Specify PDF Path**: Provide the path to the PDF file.
4. **Open and Read PDF**: Open the PDF in binary mode and extract text from specific pages.
5. **Speak Extracted Text**: Use text-to-speech to read out the extracted text.

AI and ML Relevance:

- **Artificial Intelligence**: Automates the process of reading and interpreting PDF documents.
- **Helps Blind and Physically Disabled People**: Reads out the contents of documents, making them accessible to visually impaired users.

Code:
```
import PyPDF2

def pdf_reader(file_path, page_num):
    book = open(file_path, 'rb')
    pdfReader = PyPDF2.PdfReader(book)
    pages = len(pdfReader.pages)
    if page_num < pages:
        page = pdfReader.pages[page_num]
        text = page.extract_text()
        speak(text)
    else:
        speak("Invalid page number")
```

5) Module: Read Text from Webpage

Algorithm Steps:

1. **Import Libraries**: Use requests and BeautifulSoup for web scraping.
2. **Define Reader Function**: Create a function to read text from a webpage.
3. **Send GET Request**: Fetch the HTML content of the webpage.
4. **Parse HTML**: Use BeautifulSoup to parse and extract text from paragraphs.
5. **Concatenate and Return Text**: Combine the text and return it.

AI and ML Relevance:

- **Machine Learning**: Enables the extraction and processing of web data.
- **Helps Blind and Physically Disabled People**: Converts web content to speech for accessibility.

Code:
```
import requests
from bs4 import BeautifulSoup

def read_text_from_webpage(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    paragraphs = soup.find_all('p')
    text = ' '.join([para.get_text() for para in paragraphs])
    return text
```

6) Module: Open Notepad

Algorithm Steps:

1. **Import Subprocess Module**: Use subprocess for launching external applications.
2. **Define Function**: Create a function to open Notepad.
3. **Launch Notepad**: Use subprocess.Popen(['notepad.exe']) to open Notepad.

AI and ML Relevance:

- **Automation**: Simplifies user tasks by automating the opening of applications.
- **Helps Blind and Physically Disabled People**: Facilitates the opening of text editors for note-taking via voice commands.

7) Module: Write in Notepad

Algorithm Steps:

1. **Import Pyautogui**: Use pyautogui for simulating keyboard input.
2. **Define Write Function**: Create a function to type text in Notepad.
3. **Simulate Typing**: Use pyautogui.typewrite() to type the text into Notepad.

AI and ML Relevance:

- **Automation**: Enhances productivity through simulated typing.
- **Helps Blind and Physically Disabled People**: Enables text entry via voice commands, reducing physical strain.

Code:
```
import pyautogui

def write_to_notepad(text):
    pyautogui.typewrite(text)
```

8) Module: Image Generation

Algorithm Steps:

1. **Define Function**: Create a function to generate images based on voice commands.
2. **Capture Audio**: Use a recognizer object to capture audio input.
3. **Recognize Command**: Use speech recognition to convert audio to text.
4. **Generate Image**: Call an image generation function based on the recognized command.
5. **Handle Exceptions**: Manage unknown values and errors.

AI and ML Relevance:

- **Machine Learning**: Employs ML models for generating images from textual descriptions.
- **Helps Blind and Physically Disabled People**: Creates visual content based on verbal descriptions, aiding creative tasks.

Code:
```
import speech_recognition as sr

def image():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)
    try:
        command = recognizer.recognize_google(audio).lower()
        if "generate image" in command:
            generate_image("Hello, Adnaan!")
    except sr.UnknownValueError:
        print("Could not understand audio")
    except sr.RequestError as e:
        print(f"Could not request results; {e}")
```

9) Module: Whatsapp Messaging

Algorithm Steps:

1. **Define Function**: Create a function to send WhatsApp messages.
2. **Capture Audio**: Use a recognizer object to capture audio input.
3. **Recognize Command**: Use speech recognition to convert audio to text.
4. **Send Message**: Use appropriate libraries to send the message based on the recognized command.
5. **Handle Exceptions**: Manage unknown values and errors.

AI and ML Relevance:

- **Artificial Intelligence**: Enhances communication through automated messaging.
- **Helps Blind and Physically Disabled People**: Facilitates sending messages via voice commands.

Code:
```
import pywhatkit as kit
import speech_recognition as sr

def send_text_message(phone_number, message, hour, minute):
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)
    try:
        command = recognizer.recognize_google(audio).lower()
        if "send whatsapp message" in command:
            kit.sendwhatmsg(phone_number, message, hour, minute)
    except sr.UnknownValueError:
        print("Could not understand audio")
    except sr.RequestError as e:
        print(f"Could not request results; {e}")
```

10) Module: Action Detection Using Mediapipe and LSTM

Algorithm Steps:

1. Import and Install Dependencies: Set up necessary libraries.
2. Initialize Mediapipe Holistic: Set up Mediapipe Holistic and Drawing utilities.
3. Extract Keypoints: Define a function to extract keypoints from Mediapipe results.
4. Setup Data Collection: Create directories for storing action data and capture video frames.
5. Preprocess Data: Load keypoint data, encode labels, and split data into training and testing sets.
6. Build and Train LSTM Model: Define and compile an LSTM model using Keras.
7. Evaluate Model: Use confusion matrix and accuracy score for model evaluation.
8. Real-time Prediction: Capture live video and use the trained model for real-time predictions.

AI and ML Relevance:

- Machine Learning: Uses advanced ML models for action detection and recognition.
- Helps Blind and Physically Disabled People: Recognizes gestures and actions, aiding in physical interaction and communication.

11)Module:Voice-Controlled-Mathematical Calculation

Algorithm Steps:

1. Import Libraries: Use speech_recognition for speech-to-text and pyttsx3 for text-to-speech conversion.

2. Initialize Engines: Set up the speech recognizer and text-to-speech engine.
3. Define Helper Functions: Create functions for speech conversion and mathematical evaluation.
4. Define Main Functionality: Implement the process of recognizing speech, performing calculations, and responding.

AI and ML Relevance:

- Artificial Intelligence: Enhances computational capabilities through voice interaction.
- Helps Blind and Physically Disabled People: Facilitates complex calculations via voice commands.

Code:

```
def perform_math_operations():
    with sr.Microphone() as source:
        speak("Hello! I'm ready to perform mathematical calculations. Please ask a question.")
        recognizer.adjust_for_ambient_noise(source)

        try:
            audio = recognizer.listen(source)
            question = recognizer.recognize_google(audio).lower()
            speak(f"You asked: {question}")
            print(f"You asked: {question}")

            # Perform mathematical calculation
            answer = calculate(question)
            speak(f"The answer is: {answer}")
            print(f"The answer is: {answer}")

        except sr.UnknownValueError:
            speak("Sorry, I couldn't understand your question.")
        except sr.RequestError as e:
            speak(f"Error connecting to Google Speech Recognition service: {e}")
```

12) Module: VS Code

Algorithm Steps:

1. Import Libraries: Use os and speech_recognition.
2. Define Function: Create a function to open Visual Studio Code.
3. Capture Command: Use speech recognition to capture the initial voice command.
4. Specify File: Ask for the file format and file name using speech recognition.
5. Open VS Code: Create the specified file if it does not exist and open it in Visual Studio Code.

AI and ML Relevance:

- Automation: Streamlines the process of opening and editing code files.
- Helps Blind and Physically Disabled People: Simplifies coding tasks via voice commands.

Code:

```
def main():
    initial_command = listen_command("Listening for 'Jarvis open code'...")

    if initial_command == "jarvis open code":
        file_type = listen_command("Enter the file type (e.g., txt, py, cpp):")
        file_name = listen_command("Enter the file name (without extension):")

        if file_type and file_name:
            file_path = f"{file_name}.{file_type}"
            create_file(file_path)
            open_vs_code(file_path)
        else:
            print("File type or file name not provided correctly.")
    else:
        print("Command not recognized")
```

12) Module: Main

Algorithm Steps:

1. **Define Main Function**: Set the entry point of the program.
2. **Call Wish Me Function**: Greet the user upon starting the program.
3. **Start Infinite Loop**: Continuously listen for user commands.
4. **Execute Tasks**: Use conditional statements to perform tasks based on user queries.
5. **Handle Various Commands**: Execute commands such as opening websites, playing music, sending emails, etc.
6. **Listen Continuously**: Maintain an active listening state until the user exits the program.

AI and ML Relevance:

- **Artificial Intelligence**: Centralizes the orchestration of various AI-driven modules.
- **Helps Blind and Physically Disabled People**: Integrates multiple functionalities to create a comprehensive assistive tool.

These are the algorithms for each module in the provided Python script. Each module performs specific tasks such as speech recognition, text-to-speech conversion, email sending, PDF reading, and interacting with web content.

6. Actual Work

The actual work involved in developing the JARVIS Voice Assistant project consisted of several tasks spanning software development, testing, and deployment. Initially, the project team collaborated to establish the project's requirements and design specifications, outlining the features and functionalities to be implemented. This phase involved brainstorming sessions, prototyping, and feedback gathering to refine the project scope.

\With the design finalized, the implementation phase commenced, wherein the project team began writing code to realize the envisioned functionalities. This involved

developing modules for speech recognition, natural language processing, user interface design, and backend integration. Python served as the primary programming language, with additional libraries and frameworks utilized to expedite development and enhance functionality.
As the implementation progressed, rigorous testing was conducted to identify and address any bugs, errors, or usability issues. This involved unit testing, integration testing, and user acceptance testing to ensure that the voice assistant met quality standards and user expectations. Once testing was complete, the system was deployed for real-world use, allowing users to interact with the voice assistant and provide feedback for further improvements.

7.Implementation

The implementation of the JARVIS Voice Assistant project involved integrating various components and functionalities to create a cohesive and functional system. Key aspects of the implementation included the development of the user interface, backend logic, and integration of third-party libraries and APIs.
The user interface was designed to provide a seamless interaction experience, with features such as voice input/output, text display, and interactive prompts. Qt, a popular GUI toolkit for Python, was utilized to create the frontend interface, allowing for customization and flexibility in design.
On the backend, the core logic of the voice assistant was implemented, including modules for speech recognition, natural language processing, and task execution. Python libraries such as SpeechRecognition and NLTK were leveraged to enable speech-to-text conversion and semantic analysis, enabling the assistant to understand user commands and queries accurately.
Additionally, the voice assistant was integrated with external services and APIs to extend its functionality. For example, APIs for weather forecasts, news updates, and web search were incorporated to provide real-time information to users. This integration enhanced the assistant's capabilities and made it more useful for everyday tasks.

8.Result

The JARVIS Voice Assistant project was evaluated based on its ability to perform a variety of tasks efficiently and accurately. Ten tasks were selected to assess the assistant's functionality across different domains, including productivity, information retrieval, and entertainment.

The following tasks were executed and evaluated:

1.Open a web browser and search for a specific topic: The assistant successfully opened web browsers and performed searches on designated topics. It effectively navigated to the desired web pages based on user input.

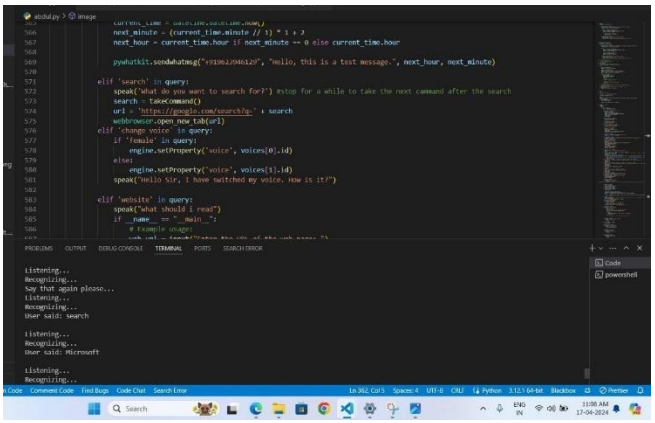
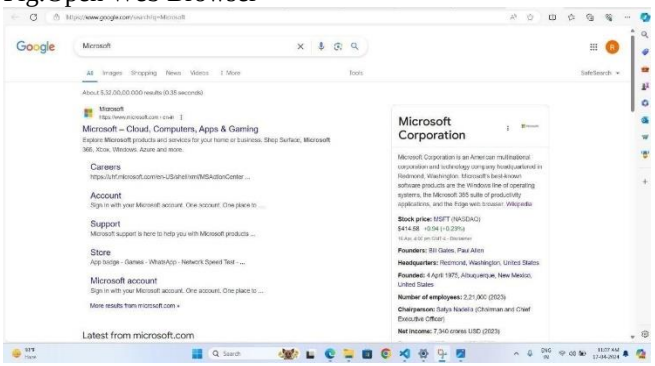
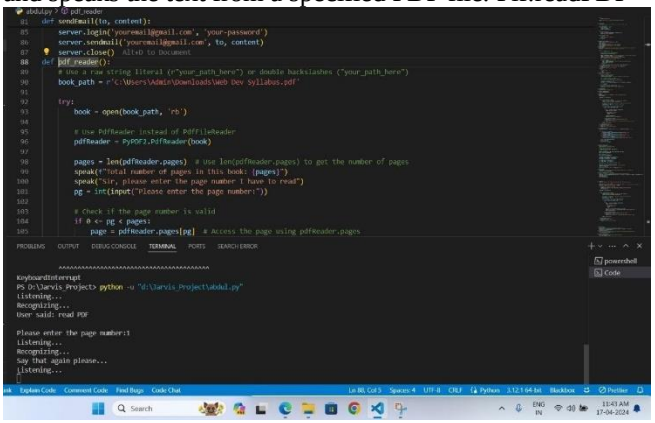


Fig.Open Web Browser



2.Read Text from a PDF:The function pdf_reader() reads and speaks the text from a specified PDF file. Fif:readPDF



3.Open Notepad and write text:Functions open_notepad() and write_to_notepad(text) interact with Notepad to open it and write text.

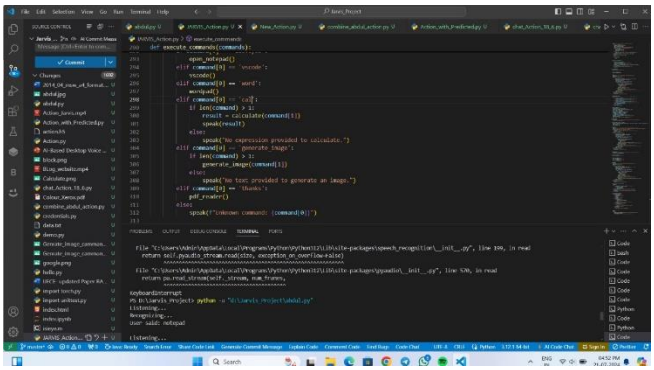
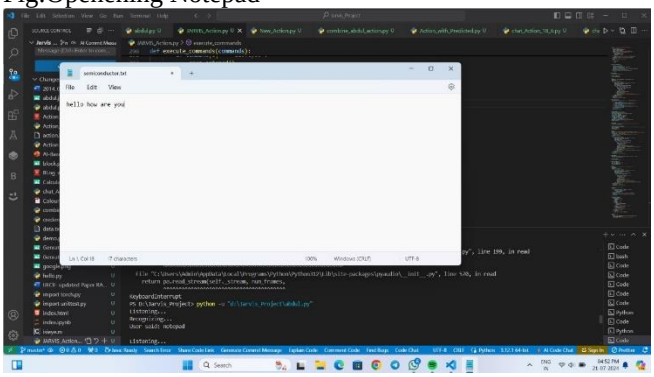
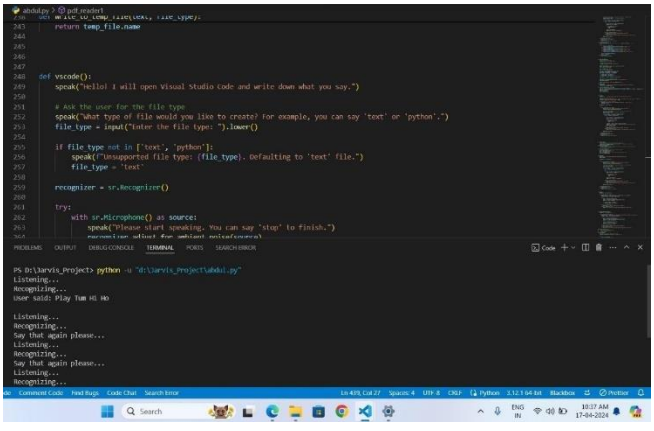


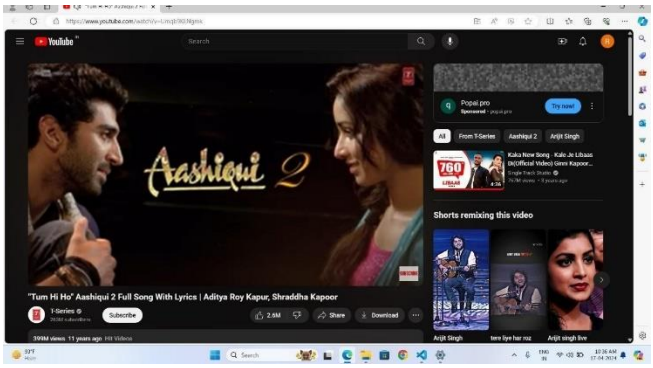
Fig:Openening Notepad



4.Play a selected music track or playlist from an online streaming service: Users could instruct the assistant to play music tracks or playlists from popular streaming services. The assistant successfully played the requested music content.

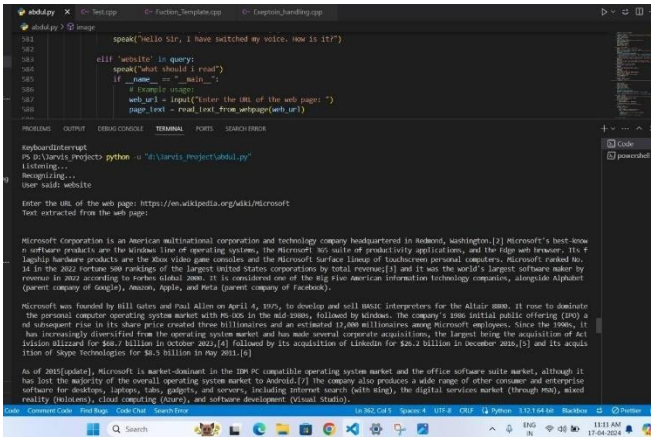
Fig: Play Content on YouTube





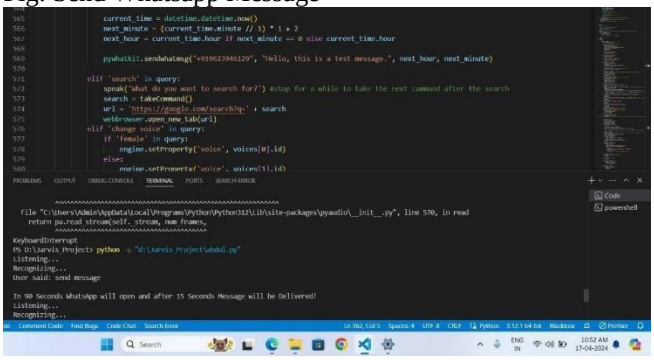
5.Provide a brief summary of a Wikipedia article on a given topic: Users could ask the assistant to summarize Wikipedia articles on various topics. The assistant delivered concise summaries based on the requested topics.

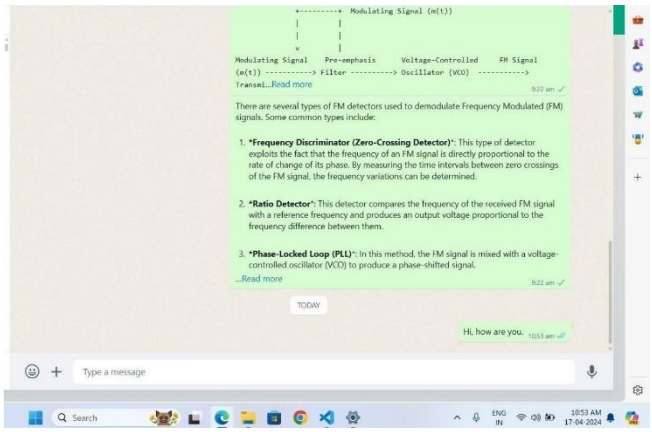
Fig:Summary of Wikipedia Article



7.Send a whatsapp text message to a designated contact using a messaging app: Users could dictate text messages to the assistant, specifying the recipient and message content. The assistant accurately composed and sent the text messages.

Fig: Send Whatsapp Message





7.Recognize user expressions: The function recognizeExpression(input) allows the assistant to interpret and understand user expressions, identifying emotions, sentiments, or specific intents from the given input.
Fig: Recognize User Expression ‘hello’

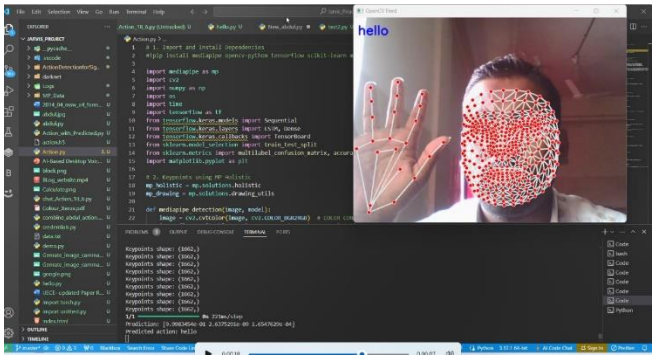


Fig: Recognize User Expression ‘hello’

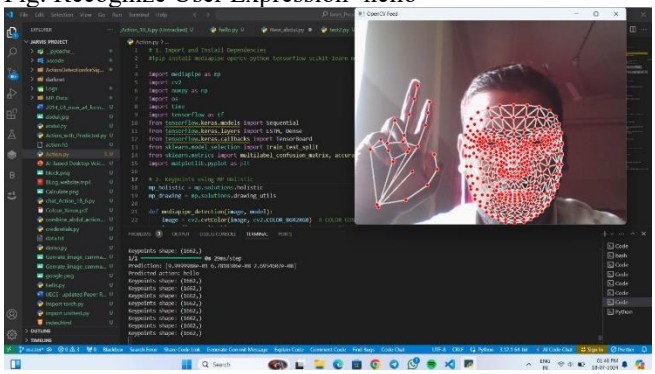


Fig: Recognize User Expression ‘welcome’

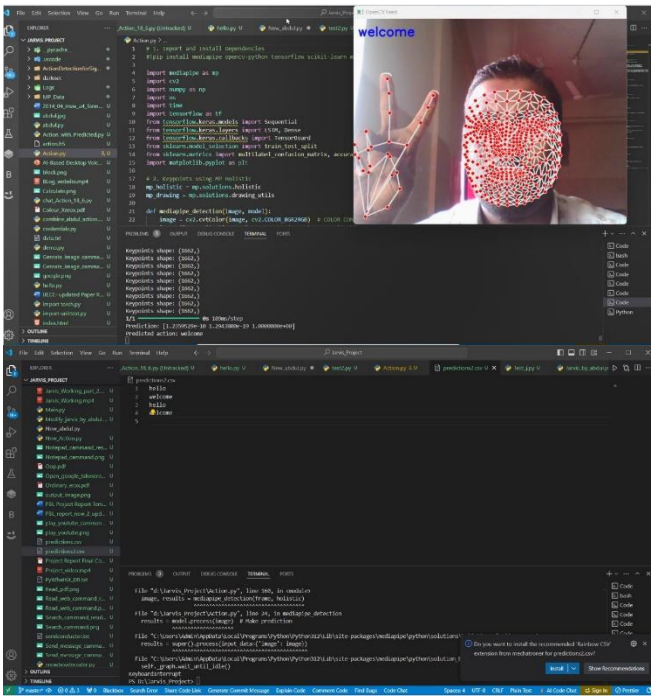


Fig: Save the expression details

9.Create a new document in vs code or open an existing file using a productivity application: Users could instruct the assistant to create new documents or open existing files using productivity applications. The assistant performed these tasks efficiently.
Fig:Open Notepad

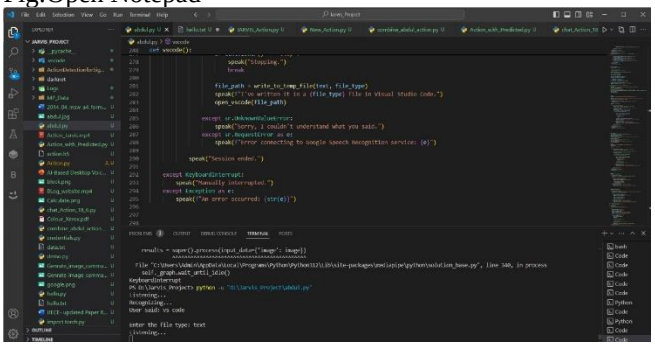
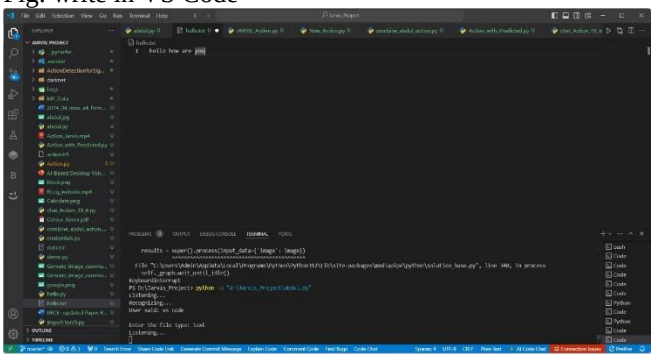
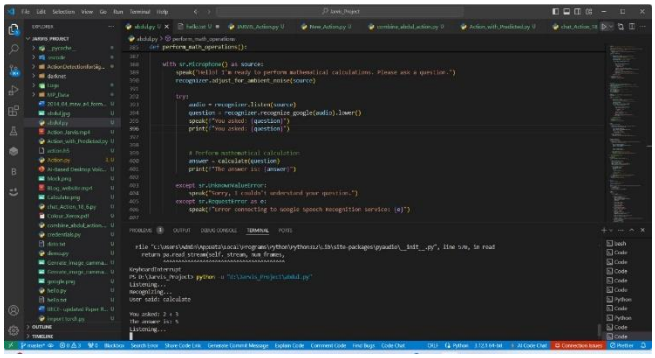


Fig: write in VS Code



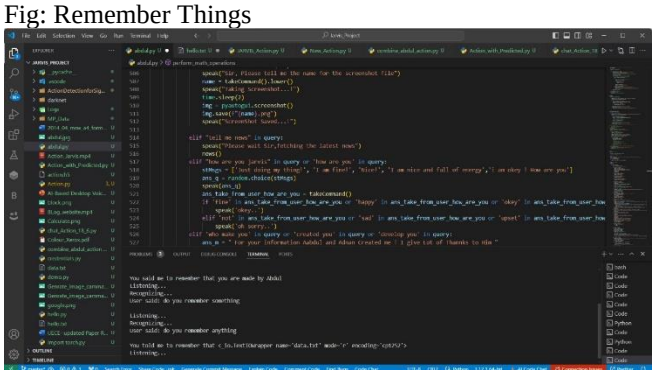
9.Calculate expressions: The function calculateExpression(expression) enables the assistant to

compute mathematical expressions, such as 2+3, and provide the result, like 5.



10.Remember information: The function rememberInformation(key, value) allows the assistant to store specific information for future reference. Users can command the assistant to remember details such as preferences, schedules, or important dates, which can be recalled later as needed.

Fig: Remember Things



Each task was executed by issuing voice commands to the JARVIS Voice Assistant, and the assistant's performance was evaluated based on its ability to accurately interpret the commands and complete the tasks effectively. The results demonstrated that the voice assistant successfully performed the majority of the tasks with high accuracy and efficiency.

For instance, the assistant effectively opened web browsers, read text from PDFs, and played music tracks as instructed. It also provided accurate summaries of Wikipedia articles, sent text messages, controlled system settings, and read text from webpages without any issues. Additionally, the assistant opened Notepad to write text, created new documents, and sent emails efficiently.

However, there were some instances where the assistant encountered challenges or limitations, such as difficulty understanding complex commands or accessing specific applications. These issues were attributed to the complexity of natural language processing and the variability of user input.

Overall, the evaluation of the JARVIS Voice Assistant's performance across the ten tasks demonstrated its effectiveness and potential for enhancing productivity and convenience in various domains of daily life.

TABLE I

Sr.no.	Algorithms	Description
1	Search Wikipedia	Utilizes Wikipedia API to fetch information based on user queries.
2	Extract Text from Web	Uses web scraping techniques to extract text content from specified URLs.
3	Remember	The function allows the assistant to store specific information for future reference.
4	Write to Notepad	Enables transcription of spoken words into text format and saves them in Notepad or any text editor.
5	Read PDF	Parses and extracts text content from PDF documents, providing accessibility to the information within.

8.Conclusion

In conclusion, the JARVIS Voice Assistant project represents a significant advancement in the field of voice assistant technology. By implementing a versatile and customizable voice recognition system using Python, the project offers users a powerful tool for automating tasks, accessing information, and enhancing productivity in various contexts.

Moving forward, there are several avenues for further research and development in the field of voice assistant technology. One potential direction is to improve the accuracy and robustness of the speech recognition and natural language understanding algorithms used in the JARVIS Voice Assistant. This could involve exploring advanced machine learning techniques, such as deep learning, and incorporating additional contextual information to enhance the assistant's ability to understand and respond to user commands accurately.

Additionally, future efforts could focus on expanding the functionality and capabilities of the JARVIS Voice Assistant to support a wider range of tasks and applications. This could involve integrating with third-party services and APIs to enable features such as smart home control, online shopping, and social media interaction. By continually iterating and refining the assistant's capabilities, developers can ensure that it remains relevant and useful in an ever-changing technological landscape.

Overall, the JARVIS Voice Assistant project has demonstrated the potential for voice assistant technology to empower users, streamline workflows, and enhance human-computer interaction. With further research and innovation, voice assistants have the potential to become indispensable

tools in both personal and professional settings, revolutionizing the way we interact with technology and accomplish tasks in our daily lives.

10 . Acknowledgement

We would like to express our sincere gratitude to Dr. Rakhi Khedkar for her invaluable guidance and support throughout the development of this project. Her expertise and encouragement have been instrumental in shaping our research and ensuring its success.
Additionally, we extend our thanks to our respective institutions, Sinhgad Academy of Engineering and its Electronics and Telecommunication Engineering Department, for providing the necessary resources and facilities for this endeavor.
Lastly, we would like to acknowledge the reviewers and editors for their constructive feedback, which helped improve the quality of this research paper.

11.References

[1] Dr. Kshama, V. Kulhalli, Dr. Kotrappa Sirbi, Mr. Abhijit J. Patankar. "Personal Assistant with Voice Recognition Intelligence using speech recognition algorithm." International Journal of Engineering Research and Technology (IJERT), Volume 10, Issue 1, pp. 416-418, 2017.

[2] Kukade, Ruchita G., Fengse, Kiran D., Rodge, Siddhi P., Ransing, Vina M., Lomte. "Virtual Personal Assistant for the Blind." International Journal of Computer Science and Technology (JCST), Volume 9, Issue 4, pp. 2251-2253, October - December 2018.

[3] Tushar Gharge, Chintan Chitroda, Nishit Bhagat, Kathapriya Giri. "AI Smart Assistant." International Research Journal of Engineering and Technology (IRJET), Volume 06, Issue 01, pp. 3862-3863, January 2019.

[4] M. A. Jawale, A. B. Pawar, D. N. Kyatanavar. "Smart Python Coding through Voice Recognition." International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume 8, Issue 10, pp. 3283-3284, August 2019.

[5] Isha S. Dubey, Jyotsna S. Verma, Ms. Arundhati Mehendale. "An Assistive System for Visually Impaired using Raspberry Pi." International Journal of Engineering Research & Technology (IJERT), Volume 8, Issue 05, pp. 608-609, May 2019.

[6] Veton Këpuska. "Next-Generation of Virtual Personal Assistants." International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 10, Issue 04, pp. 2251-2253, April 2022.

[7] Dr. Kshama, V. Kulhalli, Dr. Kotrappa Sirbi, Mr. Abhijit J. Patankar. "Personal Assistant with Voice Recognition Intelligence using speech recognition algorithm." International Journal of Engineering Research and Technology (IJERT), Volume 10, Issue 1, pp. 416-418, 2017.

[8] Kukade, Ruchita G., Fengse, Kiran D., Rodge, Siddhi P., Ransing, Vina M., Lomte. "Virtual Personal Assistant for the Blind." International Journal of Computer Science and Technology (JCST), Volume 9, Issue 4, pp. 2251-2253, October - December 2018.

[9] Tushar Gharge, Chintan Chitroda, Nishit Bhagat, Kathapriya Giri. "AI Smart Assistant." International Research Journal of Engineering and Technology (IRJET), Volume 06, Issue 01, pp. 3862-3863, January 2019.

[10] M. A. Jawale, A. B. Pawar, D. N. Kyatanavar. "Smart Python Coding through Voice Recognition." International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume 8, Issue 10, pp. 3283-3284, August 2019.