Table of Contents

AI Project: AI Assistant Final Report

Summary:

In today's tech world, the Jarvis AI Assistant is like a friendly helper that talks and does things for us. It's good at understanding what we say and can-do various tasks like checking the time and playing videos. The assistant is easy to use and can adapt to different commands, making it helpful and versatile. While it's already handy, there's a chance to make it even better in the future. The project is a starting point for beginners in AI, making talking to technology feel natural and enjoyable. The Jarvis AI Assistant promises more exciting and better experiences with smart helpers in the future.

Abstract:

This project introduces the Jarvis AI Assistant, a friendly helper that understands and does tasks based on what you say. Using speech recognition, it listens to your voice and responds by doing things like checking the time, playing videos, or opening websites. Jarvis is made to be easy to use and adaptable to different commands, creating a helpful and simple experience. It's a good starting point for people new to AI, making talking to technology feel natural and enjoyable.

Introduction:

In today’s tech world, we all need smart helpers to make our lives easier. That’s where the Jarvis AI assistant comes in. It’s like a digital friend that listens to what we say, understands our commands. And does tasks for us. This project combines talking abilities, understanding what we say, and doing things on the internet, all to create a helpful and easy experience for users. Jarvis aims to be more tha just a task-doer; it wants to be a friendly and smart companion in our digital lives. This project sets the foundation for the beginners in AI; however, the Project will be improved in future by adding more technologies that will make it more natural to interact. This Project will help us in the future to make our interaction with technology more natural and enjoyable.

1.Problem Formulation and System Design:

1.1 Problem Statement:

Develop a voice-controlled AI assistant for various tasks such as retrieving information, opening websites, playing YouTube videos, and providing date/time updates.

1.2 System Design:

1. Speech Recognition:

- Utilize the speech_recognition library to capture and process user voice input.
- Adjust for ambient noise to enhance voice recognition accuracy.
- Use Google Speech Recognition for converting audio to text.

2. Text-to-Speech:

- Use the pyttsx3 library for text-to-speech conversion.
- Configure the assistant's voice using a specific voice ID.

3. Web Browsing:

- Open specified websites (Google, YouTube, Facebook, Instagram, GitHub) using the webbrowser module.
- URLs are hardcoded for simplicity, but dynamic URL handling can be implemented for scalability.

4. Date and Time:

- Fetch the current date and time using the datetime module.
- Respond with the obtained date or time based on user queries.

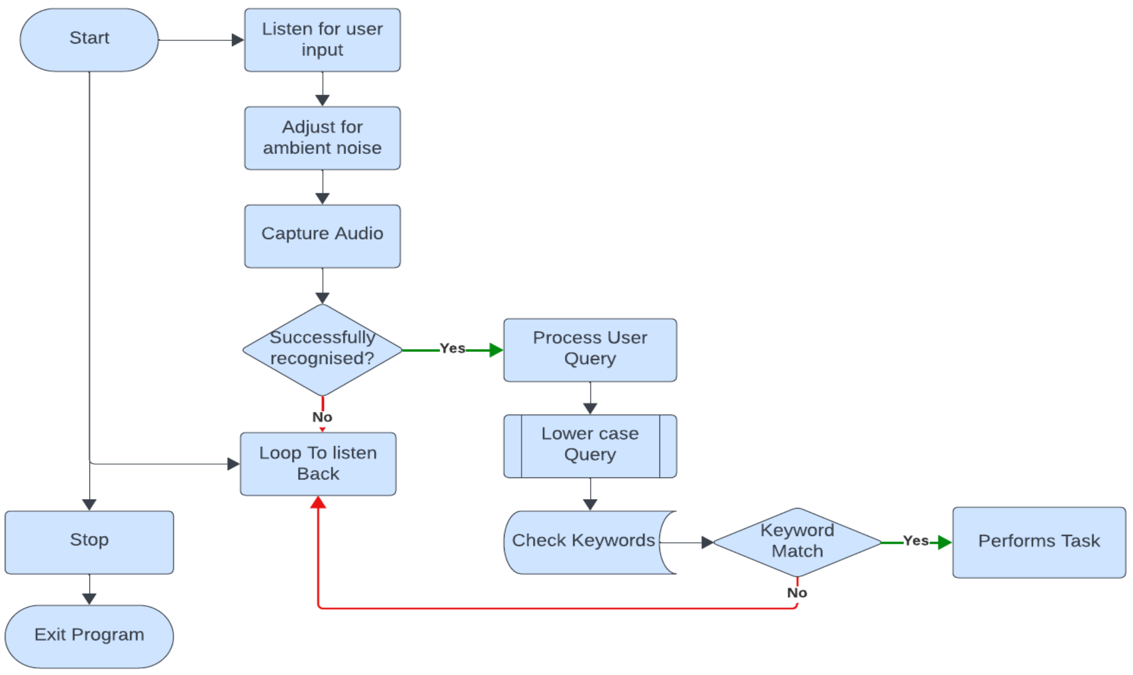
5. YouTube Video Playback:

- Use the pywhatkit library to play specified YouTube videos.
- Extract user search queries from the command and pass them to the playonyt function.

6. Exit Command:

- Recognize a specific command (e.g., "stop") to exit the assistant gracefully.
- Display a goodbye message before terminating the program.

Flowchart Diagram:



2. User Requirements and Product Specifications:

2.1 User Requirements:

1. Speech Interaction:

- The system should recognize and understand user voice commands for a natural interaction experience.
- Users should be able to initiate commands by addressing the AI assistant.

2. Text-to-Speech Output:

- The AI assistant should respond to user queries with audible text-to-speech output.
- Users should be able to hear the assistant's responses in a clear and understandable manner.

3. Web Browsing:

- Users should be able to instruct the AI assistant to open specific websites, including Google, YouTube, Facebook, Instagram, and GitHub.

- The system should provide confirmation upon successfully opening the specified websites.

4. Date and Time Information:

- Users should be able to ask the AI assistant for the current date and time.
- The assistant should respond with accurate and up-to-date information.

5. YouTube Video Playback:

- Users should be able to request the AI assistant to play specific videos on YouTube.
- The system should utilize the provided search query to play the desired video on YouTube.

6. Exit Command:

- Users should have the ability to gracefully exit the AI assistant by using a specific command.
- The assistant should provide a farewell message before terminating.

2.2 Product Specifications:

1. Speech Recognition:

- Implement the speech_recognition library to capture and interpret user voice commands.
- Utilize ambient noise adjustment for improved recognition accuracy.

2. Text-to-Speech Output:

- Use the pyttsx3 library for text-to-speech conversion.
- Configure the assistant's voice using a specific voice ID.

3. Web Browsing:

- Implement web browsing functionality using the webbrowser module.
- Provide hardcoded URLs for Google, YouTube, Facebook, Instagram, and GitHub.
- Ensure the system confirms successful website openings.

4. Date and Time Information:

- Utilize the datetime module to fetch and present the current date and time.

5. YouTube Video Playback:

- Integrate the pywhatkit library for playing specified YouTube videos.
- Extract user search queries from commands and pass them to the playonyt function.

6. Exit Command:

- Recognize the "stop" command to gracefully exit the assistant.
- Display a farewell message before terminating the program.

4. Technical Development and Literature Review

Technical Methods and Tools:

1. PYPTSX3.

- Pyptsx3 is a python library file used for importing voices for AI assistant. It contains two voices one male and one female.

2. Speech Recognition.

- Speech recognition "speech_recognition" is a python library which is used for voice commands and automation. The speech recognition function will help us recognize user voice coming from user.

3. Text to Speech.

- The function will make use of pyttsx3 library. The assistant voice will call by using assistant id stored in pyttsx3 and the speak function will be used to convert text to speech.

4. Main Execution.

- Main execution will work as a brain of assistant. It will respond to user asked tasks according to data given to it.

Literature Review.

1) PYPTSX3:

- PYPTSX3 is a Python library designed for importing voices into AI assistants. It offers two distinct voices, one male and one female.

- Voice plays a crucial role in user interaction with AI assistants. The availability of diverse voices enhances the user experience, making interactions more engaging and personalized.
- Integration of PYPTSX3 allows developers to incorporate voice diversity, catering to different user preferences and creating a more inclusive and adaptable AI assistant.

<u>2) Speech Recognition:</u>

- Speech Recognition is a Python library that facilitates voice commands and automation. It provides functions for recognizing and interpreting spoken language, enabling seamless user interaction through voice input.
- Speech recognition is pivotal for enabling hands-free and natural communication with AI assistants. It enhances accessibility and convenience for users who may prefer verbal commands over traditional input methods.
- Integration of Speech Recognition enables the AI assistant to comprehend and respond to spoken commands, broadening the scope of user interactions and improving overall usability.

<u>3) Text To Speech (TTS):</u>

- TTS is a crucial function within the assistant, leveraging the pyttsx3 library. This function utilizes an assistant ID stored in pyttsx3, and the 'speak' function converts text into speech.
- TTS is fundamental for enabling the AI assistant to communicate verbally with users. The ability to convert text to speech enhances user engagement and accessibility, particularly for individuals with visual impairments.
- The TTS function ensures that the AI assistant can convey information and responses audibly, creating a more natural and interactive user experience.

<u>4) Main Execution:</u>

- The Main Execution serves as the brain of the AI assistant, responsible for interpreting user queries and executing relevant tasks based on the provided data.
- The main execution module is the core component that determines the assistant's responsiveness and functionality. It acts as the decision-making center, orchestrating actions in response to user requests.
- This module plays a pivotal role in executing various tasks, ranging from answering queries to performing complex operations. It is essential for the overall effectiveness and utility of the AI assistant.

Future Enhancements:

1. While the current implementation is functional, improving speech recognition accuracy and expanding the range of tasks the assistant would enhance its utility.
2. Integration of natural language can make the way we talk to machines more natural and refined.
3. Develop a continuous learning system where the AI assistant learns from new data and user interactions.
4. Integrate neural network models for voice emotion recognition.
5. Utilize machine learning algorithms to analyze user data, such as search history and preferences to provide more personalized recommendations.
6. Explore self-improving algorithms that allow the AI assistant to optimize its performance itself.
7. Train the AI model to support multiple languages, enabling a more inclusive user experience.

Implementation Plan:

Step 1: Install Required Libraries

Ensure that the necessary libraries are installed. You can install them using the following commands:

- pip install pyttsx3.
- pip install SpeechRecognition.
- pip install pywhatkit.
- pip install webbrowser.
- pip install pytz.

Step 2: Set Up Text-to-Speech (TTS) Voice

- Identify the desired voice ID for the assistant in the voice_id variable within the speak function.
- Make sure the chosen voice ID corresponds to an available TTS voice on your system.

Step 3: Run the Main Loop

Execute the script by running the provided Python code in a Python environment.

Step 4: Interact with the AI Assistant

- The AI assistant will continuously listen for your voice input.
- Greet the assistant with "hello" to trigger a response.

- Ask for the time or date to receive the current time or date.
- Use keywords like "google," "youtube," "facebook," "instagram," or "github" to open corresponding websites.
- Say "play" followed by a search query to play a video on YouTube.
- To stop the assistant, say "stop."

Step 5: Observe the Output

- Monitor the console output to see the recognized user input and the AI assistant's responses.
- Check if the web browser opens the specified websites and if PyWhatKit plays the requested YouTube video.
- Step 6: Troubleshooting (if needed)
- If the assistant doesn't respond or the TTS voice is not correct, revisit the TTS voice setup.
- Ensure that the required libraries are installed and up-to-date.
- Check for any error messages in the console output and address them accordingly.

Python Pseudo Code:

```
import pyttsx3

import speech_recognition as sr

import webbrowser

from datetime import datetime

import pywhatkit


function speak(text):

    initialize pyttsx3 engine

    set voice for the assistant

    print AI response to console

    speak the provided text


function speech_recognition():
    initialize recognizer
    use microphone as source
    adjust for ambient noise
    listen to user input
    try:
        recognize audio using Google Speech Recognition
        print recognized user input to console
        return the lowercase version of the recognized text
    except UnknownValueError:
        handle when speech recognition cannot understand audio
        return an empty string
    except RequestError as e:
        handle when there is an issue with Google Speech Recognition service
        print error message to console
        return an empty string

function open_google():
    open "https://www.google.com" in default browser

function open_youtube():
    open "https://www.youtube.com" in default browser

function open_facebook():
    open "https://www.facebook.com" in default browser
```

```
function open_instagram():
    open "https://www.instagram.com" in default browser


function open_github():
    open "https://www.github.com" in default browser


function main_execution(query):
    convert query to lowercase
    if query contains "hello":
        greet the user
    elif query contains "time":
        get current time and speak it
    elif query contains "date":
        get current date and speak it
    elif query contains "google":
        open Google in the browser
        speak "Opening Google."
    elif query contains "youtube":
        open YouTube in the browser
        speak "Opening YouTube."
    elif query contains "facebook":
        open Facebook in the browser
        speak "Opening Facebook."
    elif query contains "instagram":
        open Instagram in the browser
        speak "Opening Instagram."
    elif query contains "github":
        open GitHub in the browser
        speak "Opening GitHub."
    elif query contains "play":
        extract search query from user input
        play the searched video on YouTube using PyWhatKit
    elif query contains "stop":
        speak "Goodbye! Have a good day."
        exit the program


while True:
    get user input through speech recognition
    execute main functionality based on user input
```

Conclusion:

The Jarvis AI Assistant is designed to make talking to technology easy and fun. It listens to your voice and does various tasks, making it a handy companion. The project aims to be beginner-friendly and sets the groundwork for more exciting improvements in the future. As of my experience and knowledge in AI, the Jarvis AI Assistant would be integrated with further advanced technologies and will become even more smarter and provide better experiences with helpful digital companions.