2. INTRODUCTION

The project “Jarvis – Python Voice Assistant” is designed to create an intelligent and interactive voice-controlled system capable of performing daily tasks hands-free. The assistant listens for a wake word (“Jarvis”), processes user commands through speech recognition, and executes tasks such as opening websites, playing YouTube songs, and reading live news updates.
This project demonstrates the practical use of Python libraries such as speech_recognition, pyttsx3, requests, and web automation tools. It also integrates APIs to fetch real-time data, making the assistant more dynamic and functional. The main objective is to develop an easy-to-use, fast, and helpful AI assistant that simplifies everyday computer tasks through voice commands.
The expected outcome is a fully functional voice assistant capable of understanding natural speech, processing instructions efficiently, and responding with real-time audio feedback.

3. LIBRARIES / FRAMEWORKS USED

✓ speech_recognition
Used for converting voice input into text using Google Speech API.
✓ pyttsx3
Offline text-to-speech engine used for Jarvis’s voice responses.
✓ webbrowser
Opens websites such as Google, YouTube, and Facebook automatically.
✓ requests
Fetches real-time data from online sources like NewsAPI.
✓ musicLibrary (Custom Python Module)
Stores a dictionary of song names and their respective YouTube links for music playback.

4. SYSTEM DESIGN / METHODOLOGY

The system follows a simple yet effective workflow to perform real-time voice-controlled tasks:

The assistant continuously listens through the system’s microphone.
When the user says “Jarvis”, the wake-word detection activates the assistant.
The user then gives a command such as “open YouTube” or “play Japanese”.
The speech recognized text is processed and matched with predefined functions.
Based on the command, Jarvis executes actions such as opening websites or fetching news.
The response is given back through the text-to-speech engine for natural interaction.

User Speaks → Speech Recognition → Command Processing →
Task Execution (Website/Music/News) → Voice Output

5. IMPLEMENTATION

The implementation is done entirely in Python using voice recognition and automation libraries.
Key Implementation Steps:
1. Initialize Modules
Import speech recognition, text-to-speech, webbrowser, requests, and custom modules.
Set up microphone and engine.

2. Jarvis Startup
On running the program, Jarvis says “Initializing Jarvis...”
Starts listening for the wake word.

3. Wake Word Detection
If “Jarvis” is detected, the assistant becomes active and waits for the next command.

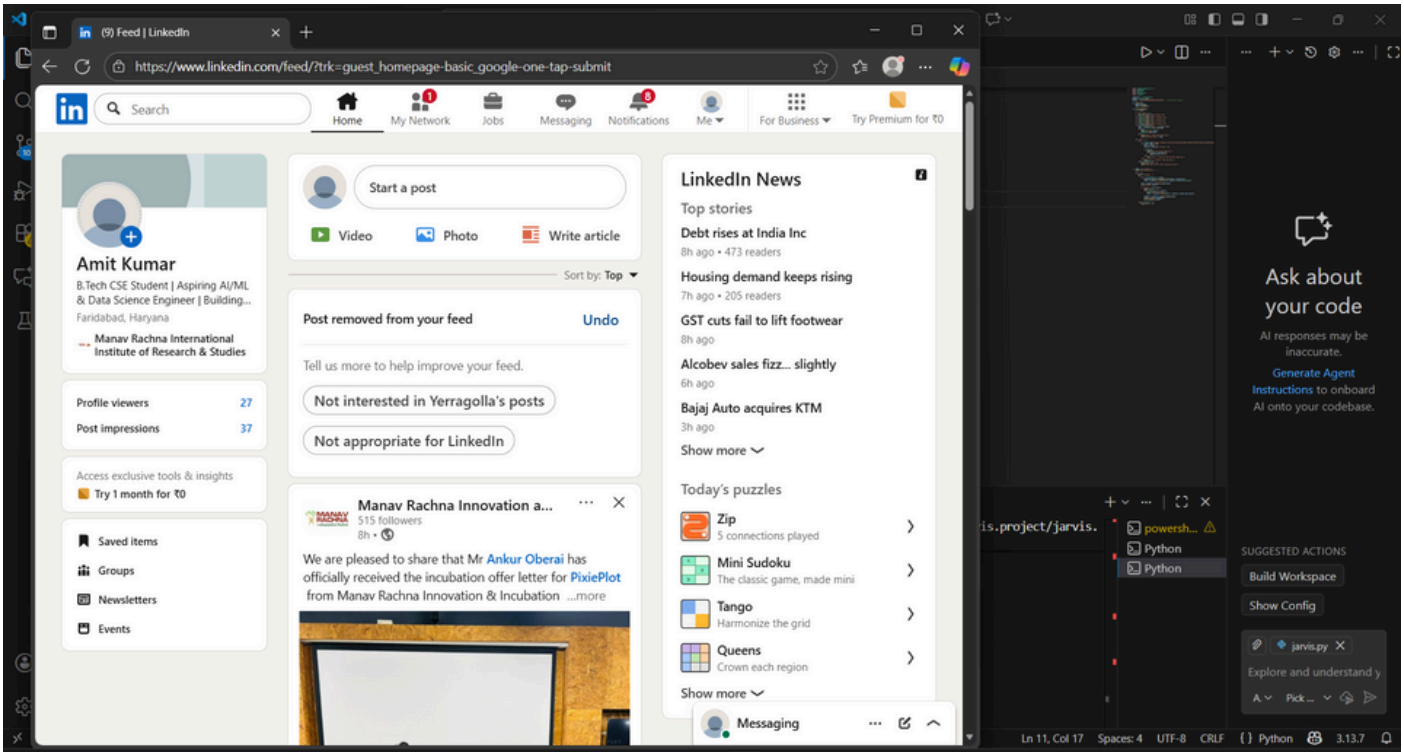
4. Command Processing
Examples:
“Open Google” → Opens google.com
“Play rap god” → Plays from musicLibrary
“News” → Fetches 5 headlines using NewsAPI
“Open YouTube” → Launches YouTube

5. Error Handling
Handles:
No voice input
Unclear commands
API failure
Unknown song name

Screenshots / Output Description

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Lenovo\Desktop\jarvis.project> & "C:/Program Files/Python313/python.exe" c:/Users/Lenovo/Desktop/jarvis.project/jarvis.py
Listening...
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\Lenovo\Desktop\jarvis.project> & "C:/Program Files/Python313/python.exe" c:/Users/Lenovo/Desktop/jarvis.project/jarvis.py
Listening...
Heard: Jarvis
Jarvis activated...
```



CONCLUSION

The Jarvis Voice Assistant project successfully demonstrates the integration of Python libraries for speech recognition, text-to-speech, automation, and API-based data fetching. The assistant can listen, understand, and execute real-time commands, making the interaction hands-free and efficient.

This project provides a strong foundation for building more advanced AI assistants in the future, with possibilities such as personalized responses, machine learning integration, home automation, and advanced NLP processing.