Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 1, 2026

JARVIS –AI INTEGRATED OPERATING SYSTEM

Mubbasher Ahmed[*1], Afzaal Akhtar[2], M.Abdul Rehman Khan[3], Noreen Khalid[4], Fazal-Ur-Rehman[5], M. Haseeb[6]

[*1,2,3,4,5,6]*School of Computer Information Technology, Beaconhouse National University*

DOI: https://doi.org/10.5281/zenodo.18629032

Keywords

Article History
Received: 01 November 2025
Accepted: 20 December 2025
Published: 03 January 2026

Corresponding Author: *
Mubbasher Ahmed

Abstract
This paper contains the fundamentals for implementing a voice assistant "Jarvis" to automate functions of windows operating systems. Jarvis was developed using python, using Ollama GPT-oss 120B as pre-trained model, the assistant provides real time voice assistance, and has the ability to make decisions based on user's request.

INTRODUCTION

Voice assistant's have been here since a long time, Samsung's Bixby and Apple's Siri being just some examples. However, the initial versions of these voice assistants had limited access to the system hardware, gradually gaining control of it. Nevertheless, these assistants are limited to their own platforms, desktops do not yet have an assistant with such ability. Cortana and Copilot are one of few desktop assistants but they do not have the ability to directly interact with the Operating System. Voice assistants use voice recognition, speech synthesis, and Natural Language Processing to provide services to the users, these services include: General Questions, Playing Music and Controlling the Media Player, Setting Tasks Reminder and Alarm, Texting and Calling etc. Our paper discusses how we enables our voice assistant Jarvis to perform some of these tasks on windows. Why do we need a voice assistant for windows? Majority of personal computers and laptops run on windows, and our voice assistant will provide an efficient method for performing tasks, instead of clicking and typing to make folders and files, you can verbally communicate with Jarvis and it will do it for you. This will also create a more personalized experience as Jarvis will remember your commands and can adapt from them. Furthermore, now you can also converse in your own language, lifting the language barriers which exist for performing day to day tasks.

Our Voice Assistant Jarvis can interact with the Operating System such that it can create directories, delete them, name them. It can create files, name them and delete them, moreover it can also read screen using AI vision. It can answer any general question, furthermore, it has the ability to open Google Chrome and Navigate to any website the user requests. It can switch between text mode and speech mode and has the ability to understand and reply in English and Urdu, User can also interrupt Jarvis by saying "Jarvis", Jarvis can shut down by saying "Deactivate" command or typing it.

Literature Review

Desktop voice assistants have emerged as a key application of artificial intelligence (AI), speech recognition, and natural language processing (NLP) to support hands-free, conversational interaction with personal computers. Early work framed virtual assistants as NLP-driven programs capable of interpreting spoken language and responding in a human-like way, integrating technologies such as automated speech recognition, text-to-speech, and data mining to enable more natural communication between humans and machines (Yulianto & Taurusta, 2024).

Evolution and Motivation. Voice assistants were first popularized on smartphones and laptops and then extended to home automation and smart speakers, before being adapted back to the desktop environment for richer system control (Gupta et al., 2022; Kunekar et al., 2023; Kumar, 2020). Desktop assistants now act as a bridge between users and complex digital ecosystems, motivated by the need for faster, more intuitive input than keyboard and mouse, as well as demands for accessibility and productivity (Kumar, 2025; Srivastava, 2024). Researchers highlight that AI-driven assistants can reduce human effort, save time, and make daily computing tasks more engaging and convenient (Kumar, 2025; Kunekar et al., 2023; Srivastava, 2024).

Core Technologies and Architectures. Most desktop voice assistants adopt a modular architecture centered on speech recognition, NLP, and dialogue/command execution. Typical systems use speech-to-text to capture commands, NLP to interpret intent, and task modules to control applications, perform web searches, manage files, or automate repetitive actions (Mukadam, 2025; Gowroju et al., 2024; Pandit et al., 2024; K & S, 2025; Basha, 2025). Python is frequently selected as the implementation language because of rich libraries for speech recognition (e.g., SpeechRecognition, pyttsx3), automation (e.g., pyautogui), and easy API integration (Kumar, 2025; Gupta et al., 2022; K & S, 2025; Basha, 2025; V.Vaitheeshwaran et al., 2023). Recent designs incorporate context-aware language models and adaptive command recognition to improve robustness and personalization (Mukadam et al., 2025; Singh et al., 2025; Mukadam, 2025; Wagner et al., 2025).

Functional Capabilities. Across studies, desktop assistants typically support opening and closing applications, web search, playing media, sending emails or messages, managing schedules and reminders, querying system status, and file operations (Mukadam et al., 2025; Singh et al., 2025; Mukadam, 2025; Kumar, 2025; Pandit et al., 2024; K & S, 2025; Basha, 2025; V.Vaitheeshwaran et al., 2023). Some systems extend to system settings control, screen capture, simple calculations, and integration with external web services for weather, news, or knowledge retrieval (Mukadam, 2025; K & S, 2025; Dharmani et al., 2024; Basha, 2025). Integration of advanced speech processing techniques and dialogue control allows more natural, multi-step task handling and closer alignment with mobile assistants like Siri, Google Assistant, and Alexa (Pandit et al., 2024; R, 2023).

Accessibility and User Experience. A major rationale for desktop voice assistants is improved accessibility. Voice interfaces offer hands-free operation and are especially beneficial for visually impaired or motor-limited users who struggle with traditional input devices (Gowroju et al., 2024; Kunekar et al., 2023; Basha, 2025). Studies stress intuitive, user-friendly graphical interfaces and simple command vocabularies to support both novice and expert users, with personalization and customization seen as critical for long-term acceptance (Singh et al., 2025; Kumar, 2025; R, 2023; Dharmani et al., 2024). Usability evaluations report high user engagement and perceived productivity benefits when assistants are responsive and accurate (Mukadam et al., 2025; Singh et al., 2025; Bhange et al., 2024).

Privacy, Security, and Deployment Models. A recurring theme is the trade-off between

cloud-based and offline architectures. Cloud connectivity enables powerful AI models but raises concerns about data privacy, latency, and dependence on network availability. Several works explicitly explore offline or locally-hosted assistants built in Python, emphasizing privacy, reduced latency, and control over data (Gupta et al., 2022; K & S, 2025; Kumar, 2020; Basha, 2025). Others note the need for secure authentication mechanisms and GUI-based login to restrict unauthorized system control (K & S, 2025; Basha, 2025).

Challenges and Limitations. Despite rapid progress, challenges remain in speech recognition accuracy (especially in noisy environments), multilingual support, and robust handling of ambiguous or complex commands (Mukadam, 2025; Pandit et al., 2024; Kumar, 2020; Bhange et al., 2024). Studies also identify limitations related to response time, dependency on stable internet (for cloud-based designs), restricted domain coverage, and difficulties integrating deeply with diverse desktop applications (Singh et al., 2025; Mukadam, 2025; Kumar, 2025; Gupta et al., 2022; Srivastava, 2024). Ethical and social concerns around user privacy, continuous listening, and responsible development are highlighted, particularly as assistants become more pervasive (Mane et al., 2025; Kunekar et al., 2023).

Recent Advances and Future Directions. Recent research focuses on leveraging state-of-the-art speech and language models to unify tasks such as wake-word detection, device-directed speech detection, and automatic speech recognition in a single end-to-end system, improving both accuracy and simplifying pipelines (Wagner et al., 2025). Other work aims to connect mobile and desktop ecosystems, providing seamless cross-device voice control and richer dialogue management (Pandit et al., 2024; R, 2023). Future directions include enhanced contextual understanding, emotion recognition, deeper personalization, tighter integration with productivity tools and smart environments, and expanded offline capabilities to balance usability with privacy (Mane et al., 2025; Pandit et al., 2024; R, 2023; Bhange et al., 2024; Srivastava, 2024).

Basic Workflow

When Jarvis is activated, it enters into a listening stage in which it initializes speech recognition and text to speech engines, and then it detects the input from the microphone and uses Google's Speech recognition API to convert speech to text while also simultaneously detecting language. It then processes it through a command handler and finally, executes the corresponding action using OS level operations, otherwise it forwards the command to Ollama AI model for a response, which is then spoken back to the user using Edge TTS, and then Jarvis returns back to listening mode.

Methodologies

System Architecture. Jarvis has a modular architecture, consisting of 3 layers: Input Layer, Processing Layer, and Output Layer. The input layer captures audio and then converts it into speech to text and it detects language simultaneously by using microphone, speech recognition library and Google Speech API. The Processing Layer routes the commands, executes OS Operations and handles general queries by using Command Processor, Ollama AI, and URL parser. Output layer generates speech audio and responds by using Edge TTS,

Data Collection and Pre-processing. Data Collection is done through Voice input, raw audio is captured at 16kHz sample rate and processed in 1024 byte chunks. Conversation history is also stored in a .json file as a list of message objects with role and content fields. Screenshots of screen are also captured via pyautogui and encoded as base64 PNG for vision analysis. The collected data undergoes Ambient Noise Adjustment, Query Normalization, URL cleaning and Response cleaning.

Speech Recognition Module. Jarvis uses the speech recognition python library with Google's Web Speech API, and pyaudio for capturing the audio. The audio is sent to the Google's could

API to be transcribed and ambient noise is calibrated to minimize background noise.

Natural Language Understanding Module. Jarvis implements NLUM through a keyword based intent classification system, combined with Ollama AI for complex queries. NLUM also handles preprocessing tasks such as extracting parameters from commands, normalizing URL inputs and cleaning speech recognition artifacts.

Dialogue Management Module. Dialogue management maintains conversation context and state through a .json based history system. Jarvis uses language aware system prompts, queries in specific languages are receive a concise persona, and appropriate TTS language is selected. It is possible because this module supports language mode switching. Furthermore, interruption handling is implemented, a background thread oversees microphone input during speech playback, and detection of input will interrupt Jarvis enabling for a new command.

Task Execution Module. Task execution module transforms recognized intents into system level actions by using Python's standard libraries and Windows API's for OS related tasks, general queries or Web Browser tasks.

Results

The Windows voice assistant displayed consistent and robust performance in recognizing user requests, and then implementing it, successfully implementing, and fulfilling the demanded request. The system executes tasks efficiently, moreover, it demonstrated it's adaptability and versatility across contexts and various queries of the user. The results display the ease of usability, and elevate the user productivity and experience in hand's free desktop environment.

Conclusion

This paper presents a highly intelligent voice assistant named Jarvis (Just a rather very intelligent system) based on python. Jarvis performs speech recognition, cloud based intent classification, real time web searches, real time responses, vision analysis, and task automation. It is built by using Open source Pre trained Large parameter (120B) Model Ollama, Google's Web Speech API, python libraries such as pygame, pyautogui, edgetts and pyttsx3. It's modular architecture makes it scalable, responsive, and robust. Future improvements include but are not limited to: offline speech recognition and response, wider range of languages to choose from, faster speech response time, and faster implementation time.

REFERENCES

Mukadam, Y., Baig, M., Seniwal, A., & Gharat, R. (2025). Jarvis – Voice Assistant for Desktop. *International Journal of Advanced Research in Science, Communication and Technology*. https://doi.org/10.48175/ijarsct-23710

Mane, T., Adhude, T., Bansode, K., Pimple, P., Khairnar, P., & Sarade, R. (2025). Virtual Personal Desktop Assistant. *International Journal of Innovative Science and Research Technology*. https://doi.org/10.38124/ijisrt/25jun487

Singh, A., Goel, K., Abbas, H., Izhar, M., & Rais, F. (2025). Voice Assistants: The Digital Age's Emergence of Virtual Friends. *AIJR Proceedings*. https://doi.org/10.21467/proceedings.178.19

Mukadam, D. (2025). An AI-Powered Desktop Voice Assistant for Windows. *International Journal of Scientific Research In Engineering and Management*. https://doi.org/10.55041/ijsrem45648

Gowroju, S., Kumar, S., & Choudhary, S. (2024). Natural Language Processing-Driven Voice Recognition System for Enhancing Desktop Assistant Interactions. 2024 7th International Conference on Contemporary Computing and Informatics (IC3I), 7, 1136-1141. https://doi.org/10.1109/ic3i61595.2024.10829162

Kumar, A. (2025). Developing a Desktop Voice Assistant. *International Journal of Scientific Research in Engineering and Management*. https://doi.org/10.55041/ijsrem41135

Pandit, S., Gupta, R., Gupta, S., & Tyagi, S. (2024). Desktop Voice Assistant: Leveraging the Current State-of-the-Art in Speech Processing. 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 1-5. https://doi.org/10.1109/icrito61523.2024.10522178

Gupta, U., Jindal, U., Goel, A., & Malik, V. (2022). Desktop Voice Assistant. *International Journal for Research in Applied Science and Engineering Technology*. https://doi.org/10.22214/ijraset.2022.42390

Kunekar, P., Deshmukh, A., Gajalwad, S., Bichare, A., Gunjal, K., & Hingade, S. (2023). AI-based Desktop Voice Assistant. 2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), 1-4. https://doi.org/10.1109/icnte56631.2023.10146699

R, M. (2023). Personal A.I. Desktop Assistant. *International Journal of Information Technology, Research and Applications*. https://doi.org/10.59461/ijitra.v2i2.58

K, S., & S, M. (2025). Voice Controlled Desktop Assistant. *International Journal of Scientific Research in Engineering and Management*. https://doi.org/10.55041/ijsrem53384

Wagner, D., Churchill, A., Sigtia, S., & Marchi, E. (2025). SELMA: A Speech-Enabled Language Model for Virtual Assistant Interactions. ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing, 1-5. https://doi.org/10.1109/icassp49660.2025.10890139

Kumar, L. (2020). Desktop Voice Assistant Using Natural Language Processing (NLP). *International Journal for Modern Trends in Science and Technology*. https://doi.org/10.46501/ijmtst061262

Dharmani, A., Khatpe, M., Gayake, P., Sharma, S., Ali, A., Dubey, S., Dwivedi, S., Pandey, M., Muskan, R., , S., Kulkarni, V., Kallurkar, S., Waikar, V., Patil, S., Deshpande, S., Gomathy, D., Narayana, R., Khrishna, T., & Geetha, V. (2024). AI - Voice Desktop Assistant. *International Research Journal of Modernization in Engineering Technology and Science*. https://doi.org/10.56726/irjmets51616

Bhange, A., Virutkar, P., Bijagare, S., Nemad, T., & Nandekar, S. (2024). An Approach Towards to Real Time AI Desktop Voice Assistant. *International Journal of Advanced Research in Science, Communication and Technology*. https://doi.org/10.48175/ijarsct-18518

Basha, M. (2025). Voice-Enabled Personal Desktop Assistant for System Control and Automation. *International Journal of Scientific Research in Engineering and Management*. https://doi.org/10.55041/ijsrem50673

Yulianto, D., & Taurusta, C. (2024). Artificial Intelligence Virtual Assistant Using Natural Language Processing Methods. *Journal of Information and Computer Technology Education*. https://doi.org/10.21070/jicte.v8i2.1664

Srivastava, A. (2024). Desktop Assistant using Machine Learning. *International Journal of Scientific Research in Engineering and Management*. https://doi.org/10.55041/ijsrem34251

Vaitheeshwaran, V., Reddy, V., Gumpula, M., & ReddyV. (2023). A Voice-Based Virtual Desktop Assistant Using Nlp Collaborating With Python. *Journal of Nonlinear Analysis and Optimization*. https://doi.org/10.36893/jnao.2023.v14i2.112

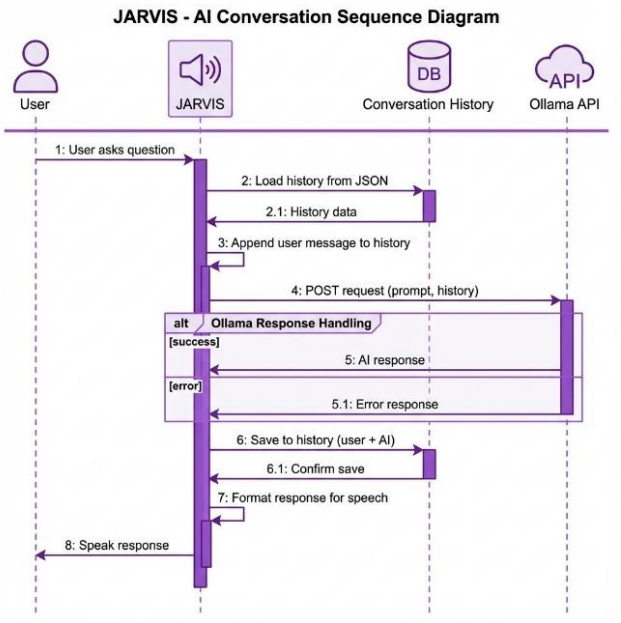
Summary Table of Key Themes

Theme	Main Points	Citations
Core technologies	Speech recognition, NLP, dialogue control, Python-based architectures	(Mukadam, 2025; Gowroju et al., 2024; Kumar, 2025; Pandit et al., 2024; K & S, 2025; Basha, 2025; V.Vaitheeshwaran et al., 2023)
Productivity & automation	Automating routine desktop tasks; hands-free control; cross-app integration	(Mukadam et al., 2025; Singh et al., 2025; Mukadam, 2025; Kumar, 2025; Pandit et al., 2024; K & S, 2025; Basha, 2025; Srivastava, 2024)
Accessibility	Benefits for visually impaired and differently-abled users; inclusive interfaces	(Gowroju et al., 2024; Kunekar et al., 2023; K & S, 2025; Basha, 2025)
Privacy & offline operation	Local processing, offline Python assistants, GUI authentication	(Gupta et al., 2022; K & S, 2025; Kumar, 2020; Basha, 2025)
Challenges & research gaps	Noise robustness, ambiguity handling, multilingual support, privacy and security concerns	(Singh et al., 2025; Mukadam, 2025; Pandit et al., 2024; Gupta et al., 2022; Kumar, 2020; Bhange et al., 2024; Srivastava, 2024)
Advanced models & future work	Integrated speech-LLM models, cross-device assistants, richer contextual and emotional understanding	(Pandit et al., 2024; R, 2023; Wagner et al., 2025; Bhange et al., 2024; Srivastava, 2024)

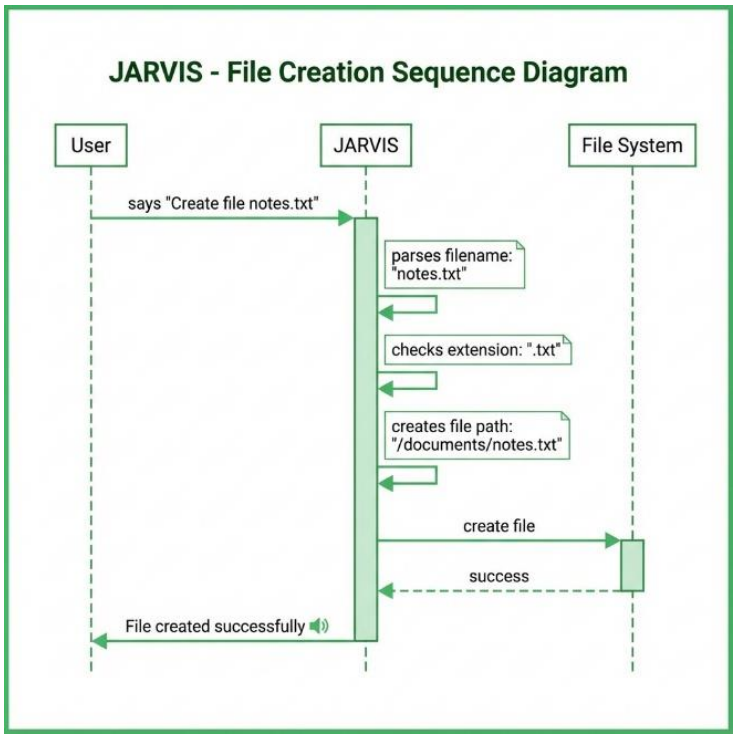
Figure 1: Key recurrent themes in desktop voice assistant research.

Figures

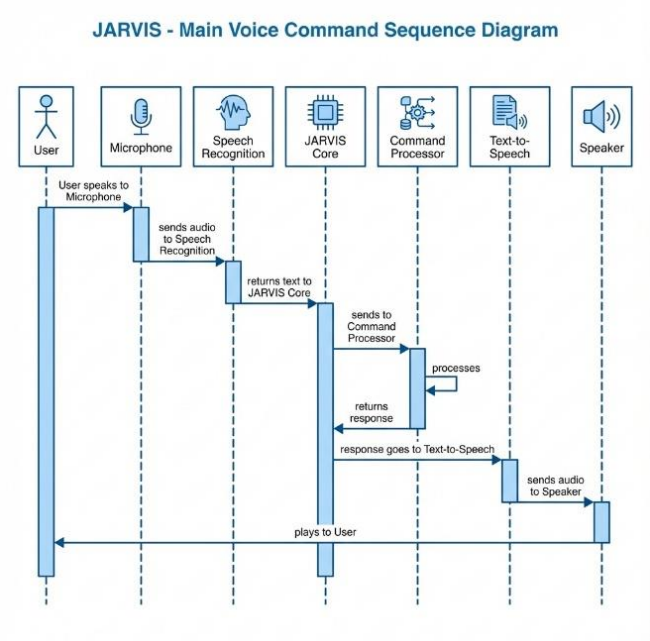
Sequence Diagrams



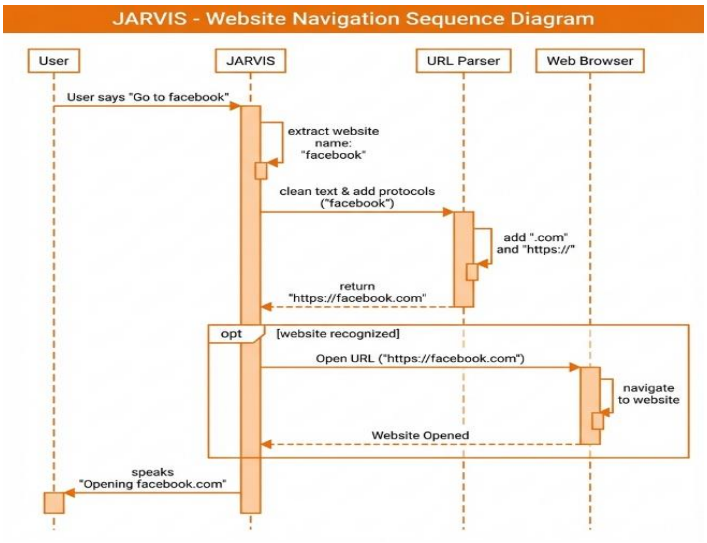
Fig(1.0)


JARVIS - File Creation Sequence Diagram
User
JARVIS
File System
says "Create file notes.txt"
parses filename: "notes.txt"
checks extension: ".txt"
creates file path: "/documents/notes.txt"
create file
success
File created successfully

Fig(1.1)

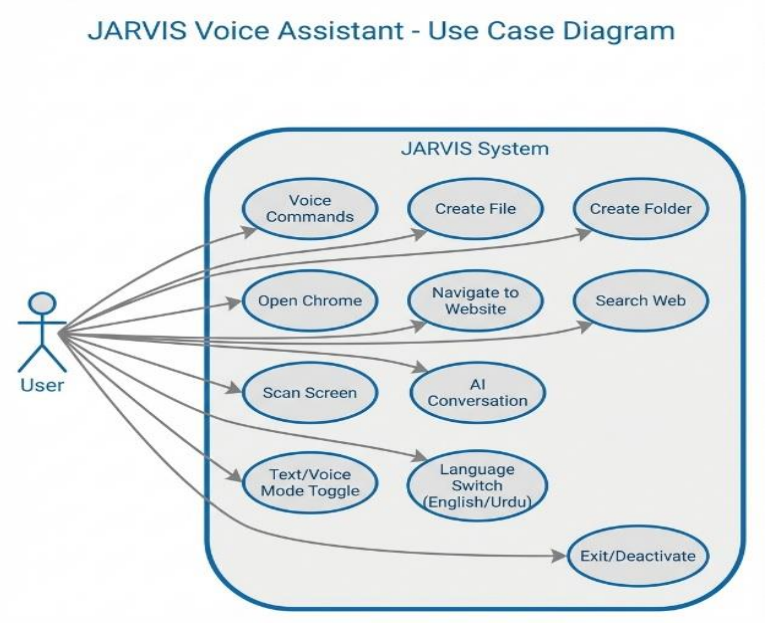

JARVIS - Main Voice Command Sequence Diagram
User
Microphone
Speech Recognition
JARVIS Core
Command Processor
Text-to-Speech
Speaker
User speaks to Microphone
sends audio to Speech Recognition
returns text to JARVIS Core
sends to Command Processor
processes
returns response
response goes to Text-to-Speech
sends audio to Speaker
plays to User

Fig(1.2)



Fig(1.3)

Use-case Diagram



Fig(2.0)