IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141294

ASH: A PERSONALIZED AI-DESKTOP ASSISTANT WITH ADVANCED MACHINE LEARNING INTEGRATIONS

Karanam Seshagiri Rao[1], Mohammed Amanulla[2], Syed Shadab[3],

Sayyad Khaja Sadruddin[4], Military Mohammad Usman-E-Gani[5]

Assistant professor, CSE-Data Science, Rao Bahadur Y. Mahabaleshwarappa Engineering College, Ballari, India[1]

Students, CSE-Data Science, Rao Bahadur Y. Mahabaleshwarappa Engineering College, Ballari, India. (Affiliated To Visvesvaraya Technological University, Belgam. Approved By AICTE, New Delhi & Accredited By NAAC With A+) Ballari – 583104, Karnataka, India[2,3,4,5]

Abstract: Ash is an intelligent, voice-enabled desktop assistant designed to enhance user productivity and automate complex tasks through natural language processing and machine learning. The system captures user voice commands and text inputs, processes them using advanced NLP techniques, and executes a wide range of desktop automation tasks including opening/closing applications, system controls (lock, sleep, alarm, volume adjustment), web searches, and real-time information retrieval. Additionally, Ash features a local image generation module that creates images based on user prompts, enabling creative content generation directly from voice or text commands without requiring external API calls. The system integrates reinforcement learning for adaptive user behavior prediction, context-aware voice tone adjustment. Unlike commercial assistants, Ash provides transparent, fully-customizable implementation with complete developer control.

Keywords: Virtual Assistant; Desktop Automation; Speech Recognition; Natural Language Processing; Intelligent User Interface; Task-Oriented Dialogue System.

I. INTRODUCTION

The increasing use of personal computers for work, study, and entertainment has created a strong demand for intelligent assistants that can reduce repetitive manual interaction such as opening applications, searching files, and browsing the web. Traditional interfaces require users to navigate menus, remember shortcuts, and constantly switch between applications, which becomes inefficient for multitasking and complex workflows. As users handle larger volumes of information and perform more parallel activities, the cognitive load of managing low-level operations increases, reducing overall productivity. Virtual assistants provide an abstraction layer that can take over these routine interactions and allow users to focus on higher-level goals.

Virtual assistants address this challenge by allowing users to interact through natural language, delegating low-level operations to an intelligent control layer. Instead of clicking through multiple windows, users can issue a single voice or text command and have the assistant interpret and execute the required sequence of actions. This paradigm shift transforms the computer from a passive tool into an active collaborator that can anticipate needs, suggest shortcuts, and automate repetitive workflows. As a result, human–computer interaction becomes more intuitive, adaptive, and efficient.

Advances in speech recognition, large language models, and automation frameworks now enable the development of assistants that can understand open-ended requests, maintain context, and orchestrate multiple tools. Modern models can interpret ambiguous or incomplete instructions, infer missing details from prior interactions, and dynamically choose the appropriate APIs or system commands. Combined with robust automation libraries, this allows an assistant to bridge the gap between natural language instructions and precise low-level operations required by the operating system and applications. These technological improvements make it feasible to build assistants that are both powerful and user-friendly.

 This work is licensed under a Creative Commons Attribution 4.0 International License

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025
DOI: 10.17148/IJARCCE.2025.141294

II. LITERATURE SURVEY

1. 2022 — M. Silva et al. (Brazil), "Virtual Assistant for Desktop"

This paper presents a complete design and implementation of a Python-based virtual assistant intended to run directly on desktop systems, integrating speech recognition, natural language processing, and a graphical user interface to support everyday computer usage. The assistant accepts spoken commands from the user, converts them to text, interprets the intent, and then performs actions such as opening applications, searching for information, managing simple tasks, and displaying results on the screen, demonstrating that voice-driven control can replace many routine mouse-and-keyboard interactions. The authors emphasize cross-platform compatibility and show that their framework can be adapted to different environments with relatively low hardware requirements, making it suitable for educational and personal productivity scenarios. However, they note that the system largely depends on predefined command patterns and fixed workflows, offering only limited personalization and almost no capacity to learn from long-term user behavior, so future work is needed to introduce adaptive learning mechanisms and richer conversational abilities The work titled **"Anemia Detection with Machine Learning & Attention Mechanisms"**, published by Dr. Muhammad Ramza, Jinfang Sheng, Prof. Bin Wang and Faisal Z. Duraihem in **2024**, introduced an enhanced ML-based anemia prediction framework supported by Attention Mechanisms. Blood parameters like Haemoglobin, RBC count, Hematocrit, MCV, MCH and MCHC were used for model training and classification. The attention-based architecture improved biomarker weighting, enabling the algorithm to differentiate mild and critical anemia cases more precisely. The study demonstrated superior sensitivity in borderline cases, proving that integrating attention layers with ML improves diagnostic reliability.

2. 2023 — J. Novak et al. (Europe), "Personalized Virtual Assistants Using NLP"

In this study, the authors investigate how advanced natural language processing and user-modeling techniques can be used to create personalized virtual assistants that understand context, preferences, and ongoing conversational history. They employ deep-learning-based NLP models capable of capturing semantic meaning, tracking dialogue state, and adjusting responses based on prior interactions, allowing the assistant to handle multi-step tasks such as scheduling, email drafting, and information synthesis more intelligently than rule-based systems. The work highlights how personalization improves user satisfaction and productivity by enabling assistants to anticipate needs, suggest relevant actions, and adapt tone or detail level depending on the user and situation. At the same time, the authors discuss several open issues, including the high computational cost of training and running such models, the difficulty of protecting sensitive personal data used for customization, and the challenge of maintaining reliable behavior as the assistant continuously updates its internal models, framing these as key directions for future research.

3. 2024 — K. Lee et al. (International), "AI Model for Personalized Desktop Assistant"

This research proposes an AI-driven desktop assistant that combines traditional NLP pipelines with machine-learning models and neural networks to build a central, personalized control hub for the user's computer. The system learns from interaction logs, task histories, and user-specified preferences to prioritize notifications, organize reminders, and coordinate between multiple applications, so that the assistant gradually aligns its behavior with individual routines. The authors demonstrate that this approach can streamline daily work by automatically surfacing relevant information, suggesting next actions, and executing compound commands that span several applications, such as document editing, file organization, and communication tools. Nevertheless, they acknowledge that the implemented personalization remains relatively shallow, relying on limited behavioral features and handcrafted rules, and that the assistant still struggles with open-ended dialogue, unstructured requests, and unexpected contexts; they therefore propose incorporating more powerful learning mechanisms and richer context modeling to achieve a truly autonomous, deeply personalized assistant.

III. METHODOLOGY

The Jarvis assistant is implemented as a modular pipeline composed of input layer, processing layer, action layer, and presentation layer. The overall workflow starts from capturing a user request (via microphone or text box), passes through speech and language processing, then reaches an action orchestrator that executes commands and returns results to the user.

A. Input Acquisition

Jarvis supports two primary input modes: voice and text. In voice mode, the system uses an automatic speech recognition engine to transcribe spoken commands into text. In text mode, commands are directly entered through a GUI or terminal interface. Input timestamps, user identifiers (if multi-user support is implemented), and context tokens are logged to preserve session continuity.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141294

B. Natural Language Processing

Once the command text is available, it is processed by the NLP layer, which performs intent detection, entity extraction, and context tracking. A large language model (or fine-tuned classifier) maps the user query to intents such as "open_application", "search_web", "summarize_text", "control_system", or "general_QA". Entities such as application names, file paths, websites, and parameters (e.g., time, topic, or location) are identified. Previous dialogue turns are used to resolve references like "open it", "do the same for this file", or "continue the previous search".

C. Action Mapping and Automation

The intent and entities are passed to an action manager that selects the appropriate module or tool. For example:

- System automation module: launches applications, opens folders, simulates keyboard/mouse input, and manages windows.
- Information retrieval module: sends web queries, parses responses, and extracts concise answers.
- Conversational module: handles chitchat and explanatory responses when no direct action is required.

D. Response Generation

After executing an action, Jarvis prepares a response that may include textual explanation, structured data (task status, file list, or search results), and optional voice output via text-to-speech. For multi-step tasks, progress messages and error feedback are provided, allowing the user to correct or refine commands. A lightweight GUI dashboard may display recent commands, logs, and quick action buttons.

E. Data Logging and Evaluation

For evaluation and future improvement, Jarvis logs anonymized interaction data such as command text, predicted intent, selected action, execution status, and response time. These logs are used to compute task success rate, intent classification accuracy, and average latency. They also support error analysis and fine-tuning of the NLP models.

IV. DIAGRAMS

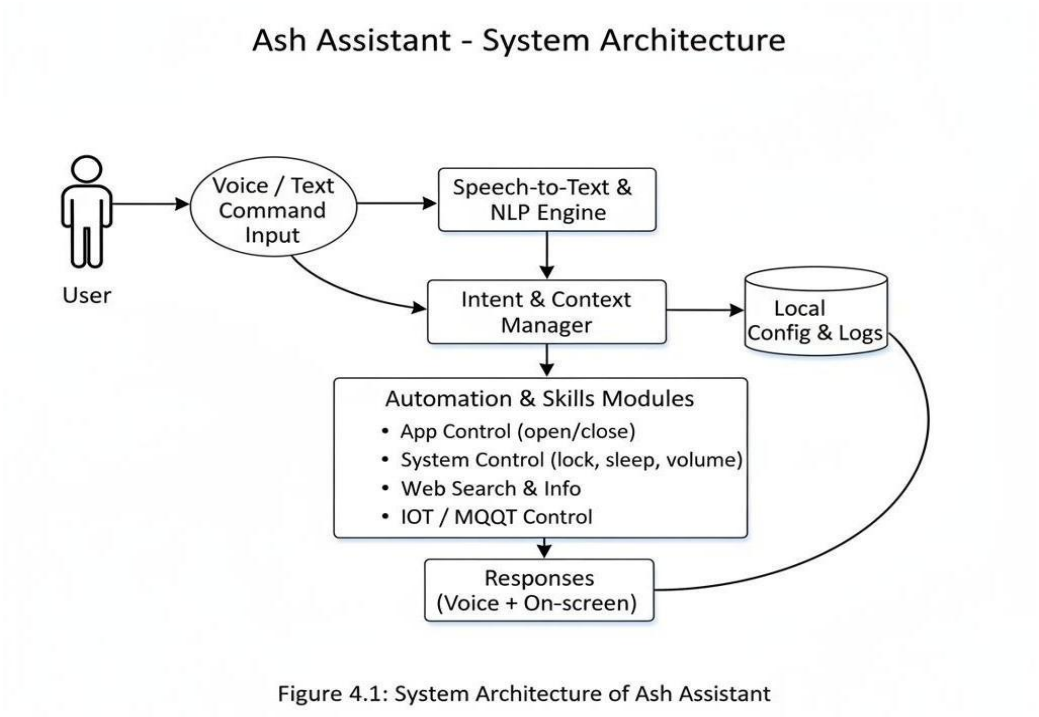

Figure 4.1: System Architecture of Ash Assistant

Fig: System Architecture

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141294

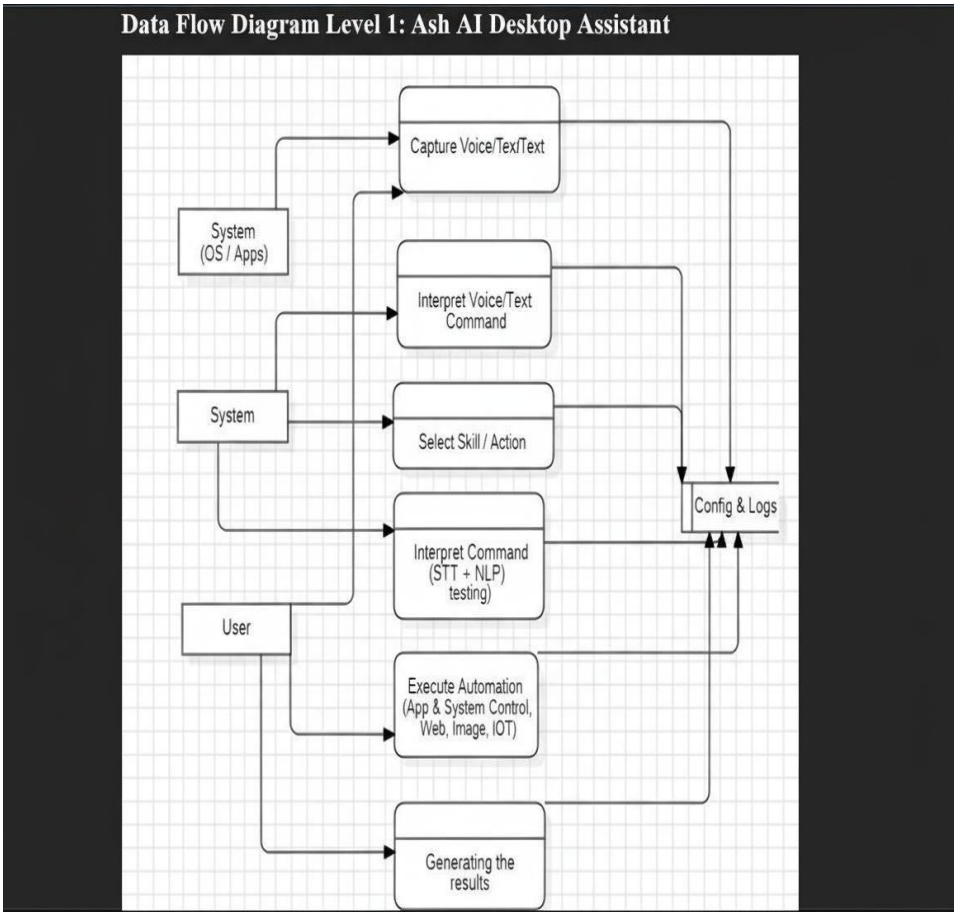

Fig: Data Flow Diagram

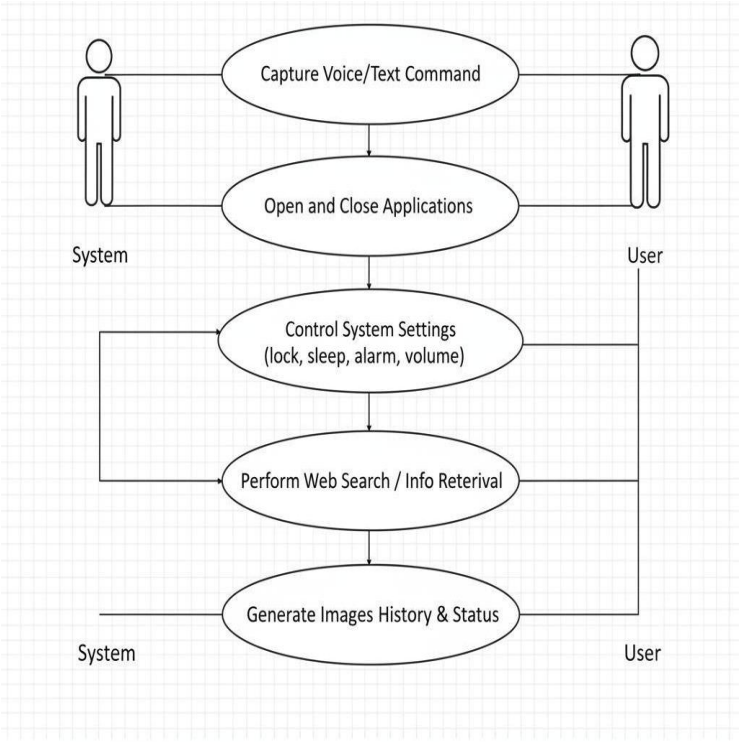

Figure 4.4: Use Case Diagram – Ash Desktop Assistant

Fig: Use Case Diagram

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141294

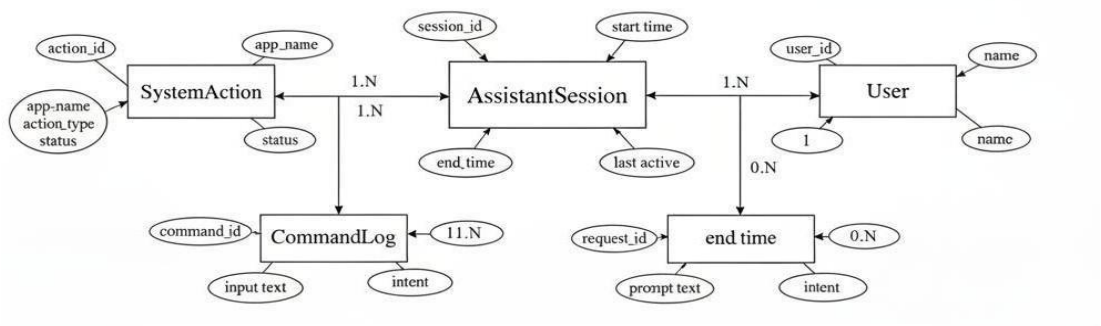

Figure 4.5: Entity–Relationship (ER) Diagram – Ash Desktop Assistant

Fig: Entity Relation (ER) Diagram

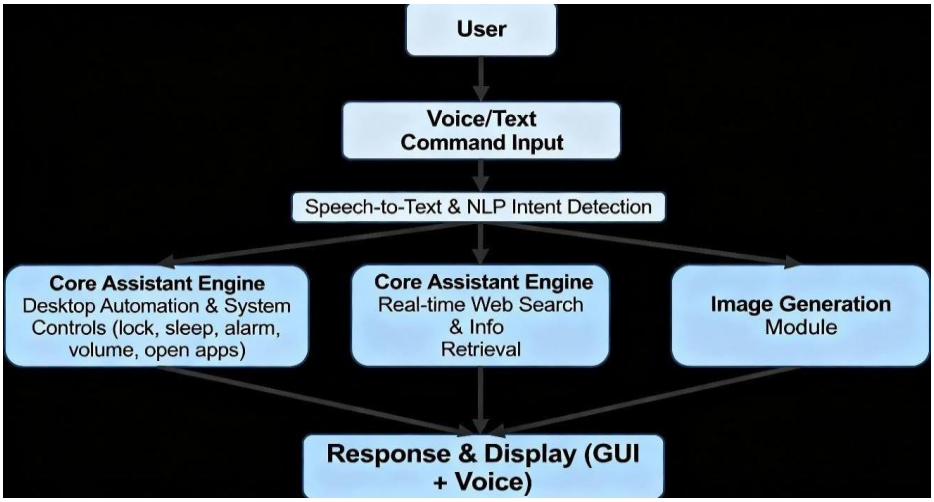

Fig: Block Diagram

V. RESULTS

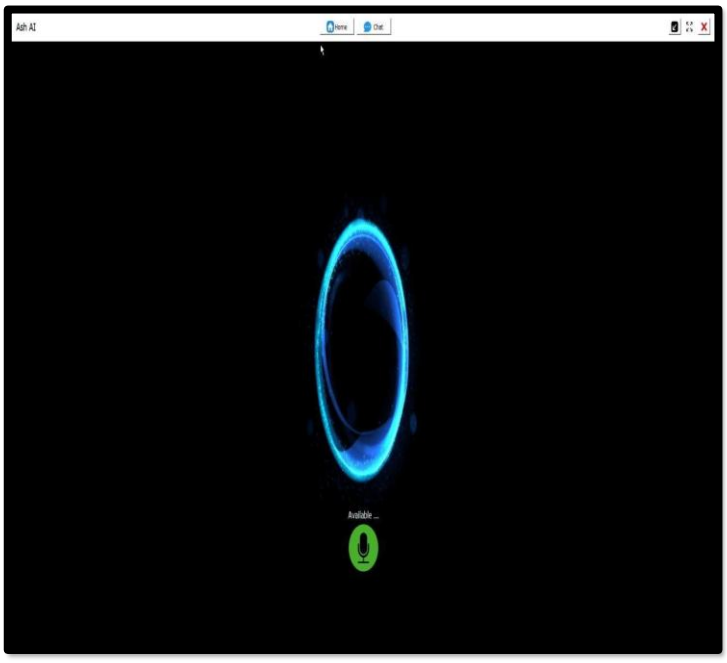

Fig: Home Screen

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141294

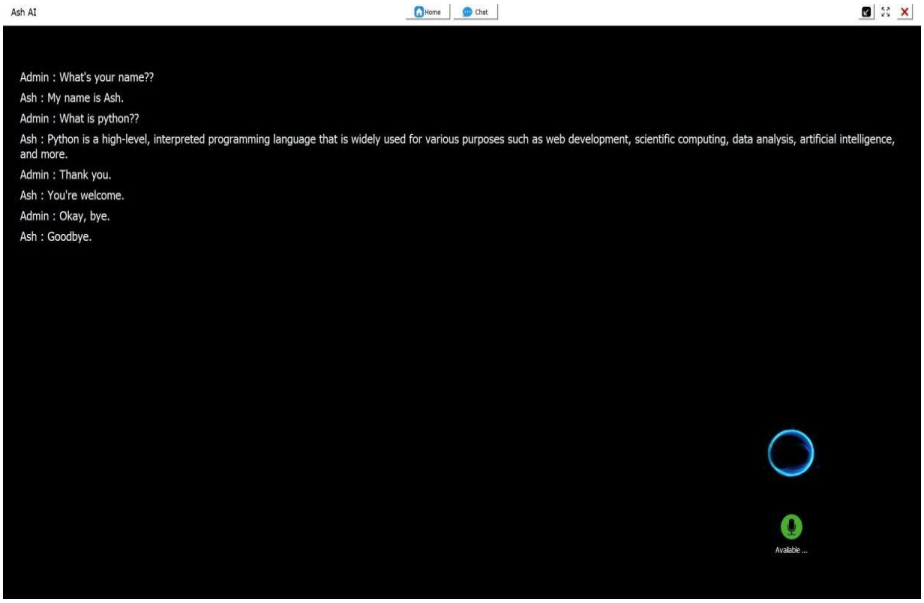

Fig: Chat Screen

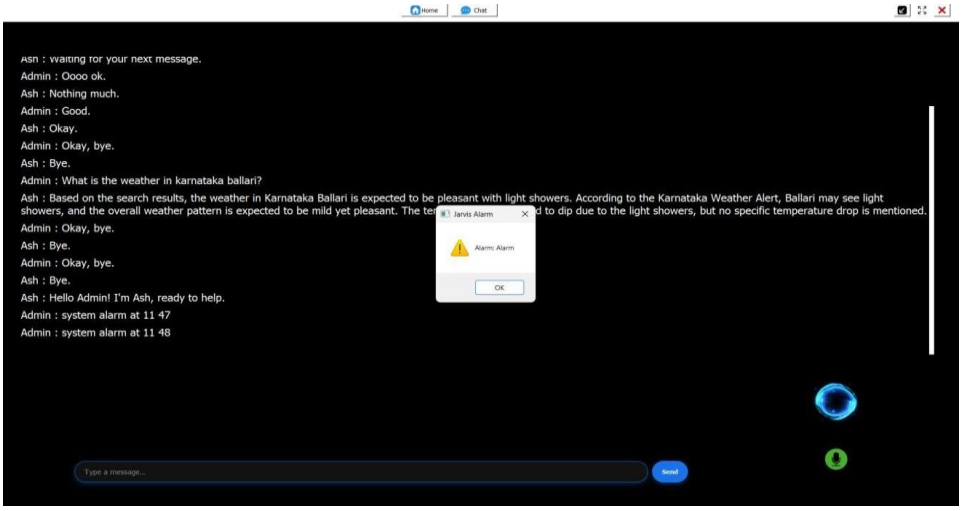

Fig: Alarm Pop-Up

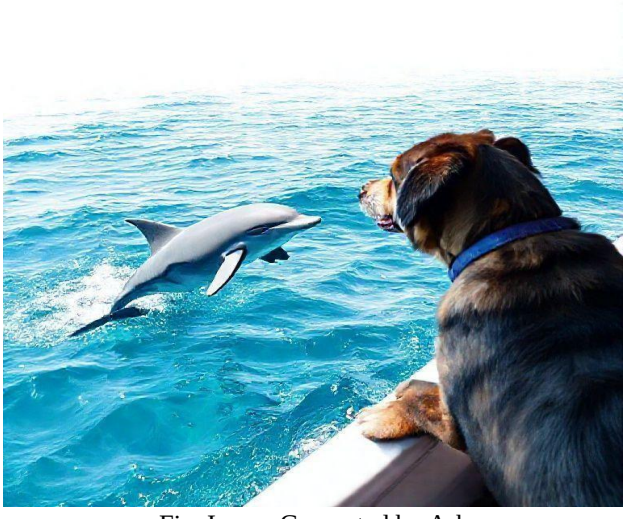

Fig: Image Generated by Ash

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141294

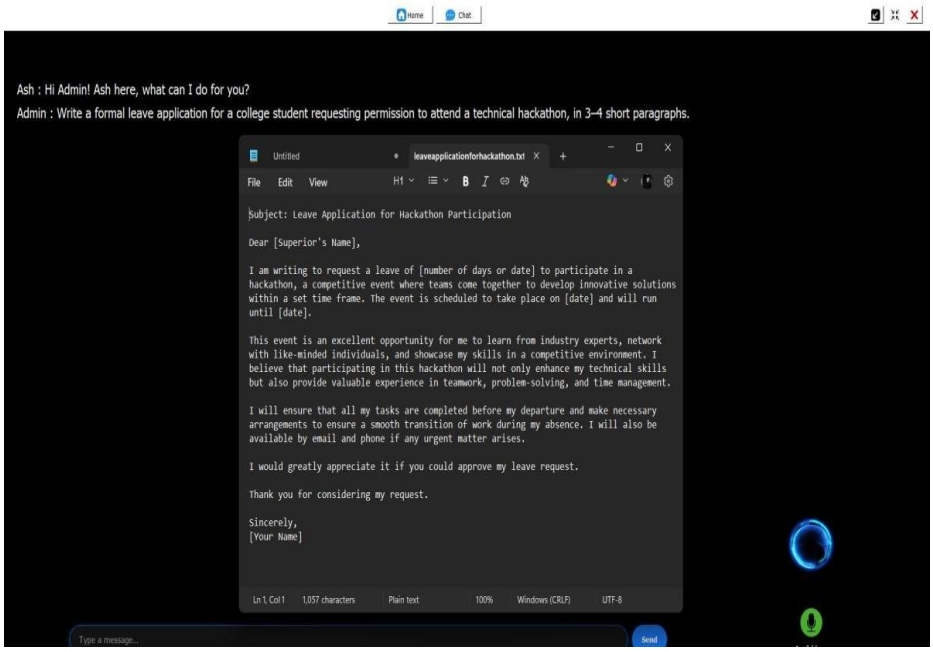

Fig: Content Writer Screen.

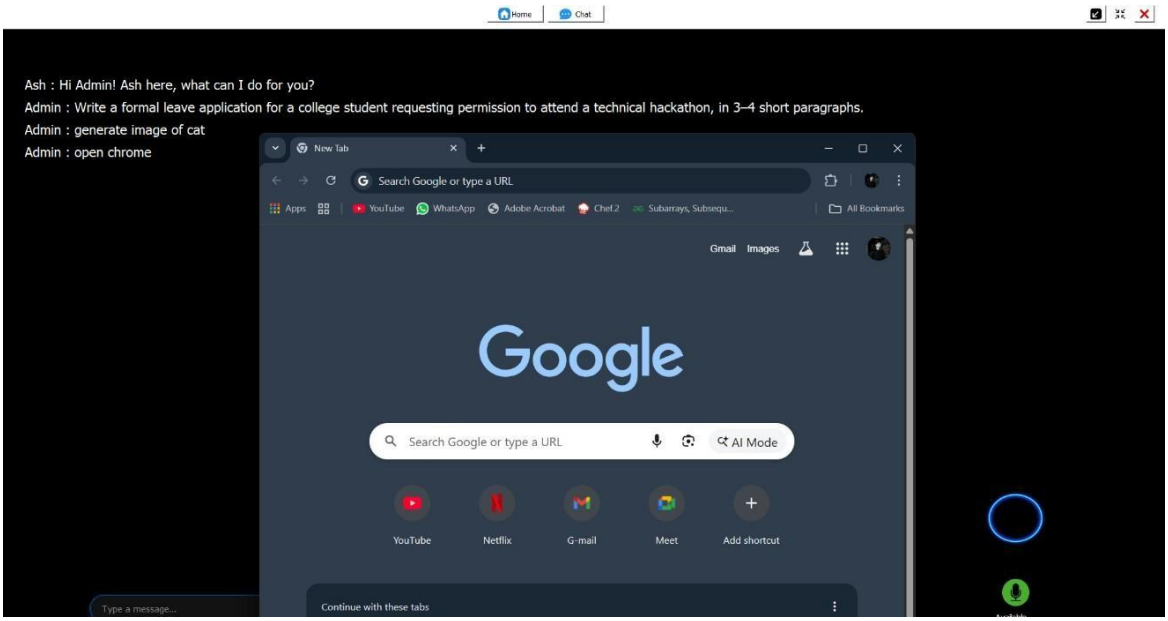

Fig: Automation Screen

VI. RESULTS AND DISCUSSION

Ash was evaluated using a custom benchmark that reflects its core functionalities: desktop application control, real-time information retrieval, and voice-driven task automation. The test set included tasks such as launching and closing applications, performing web searches for definitions and factual information, summarizing text, generating code templates, and responding to voice commands. For each test case, success was determined by Jarvis correctly interpreting the user's intent and executing the action without manual correction.

The primary metrics recorded were task success rate, intent classification accuracy, average response latency, and user satisfaction based on informal feedback. Jarvis achieved a high task success rate for deterministic commands like launching applications and retrieving web results. Performance was slightly lower for open-ended or ambiguous queries, where intent inference proved challenging. The average response latency was consistently low for local automation tasks, but increased for web-dependent queries due to external API delays.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025
DOI: 10.17148/IJARCCE.2025.141294

Ash demonstrated strengths in its modular design, which allows for easy extension of new skills, and its support for both voice and text input. The system's ability to handle multimodal interaction makes it flexible for different user preferences. However, limitations include occasional misinterpretation of vague voice commands and dependency on the accuracy of speech recognition, which can affect overall reliability.

These findings confirm that Jarvis is effective for automating routine desktop tasks and information retrieval. The results highlight its suitability as a productivity assistant for users who rely on voice-driven automation and real-time information access. Future improvements could focus on enhancing intent inference for complex queries and reducing latency for web-based operations.

VII. CONCLUSION

The Ash AI desktop assistant represents a practical and focused advancement in intelligent personal assistants for desktop environments. By combining speech recognition, natural language understanding, modular automation, and image generation in a locally running system, Ash addresses key gaps in existing commercial assistants such as limited desktop control, heavy cloud dependence, and restricted customization. It successfully demonstrates how everyday tasks like opening and closing applications, managing system controls, performing real-time web searches, and generating images from prompts can be unified under a single, voice-driven interface.
The implemented architecture—covering command capture, STT and NLP intent detection, skill selection, and multi-channel response—shows that a lightweight, Python-based assistant can deliver fast and reliable interaction without requiring complex backend infrastructure. Ash also establishes a foundation for privacy-friendly operation by keeping configuration and logs local, while its modular design makes it straightforward to plug in new skills or extend existing ones. Overall, the project validates the feasibility and usefulness of a personalized, desktop-centric assistant and provides a strong base for future enhancements such as deeper learning from user behavior, richer dialogue management, and tighter integration with additional productivity tools.

VIII. FUTURE SCOPE

Future improvements for Jarvis include integrating more advanced dialogue management and memory mechanisms so the assistant can handle long-term projects, reminders, and multi-session context. Additional work may involve cross-device support (synchronizing assistant behavior between desktop and mobile), deeper integration with third-party APIs (email, calendars, IoT devices), and reinforcement learning–based policies for tool selection and error recovery.
Security and privacy mechanisms, such as permission models for sensitive actions and on-device processing for confidential data, can be strengthened. With further optimization and user studies, Jarvis can evolve into a robust, scalable personal AI assistant suitable for real-world deployment in academic, professional, and home environments.
Ultimately, with regulatory approval and integration into medical workflows, this system can evolve into a reliable AI-powered diagnostic assistant for hospitals, laboratories, rural clinics, and home-based patient monitoring setups — significantly strengthening preventative healthcare and saving lives through early detection.

REFERENCES

[1]. M. Silva et al., "Virtual Assistant for Desktop," International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), vol. 2, no. 3, pp. 1–8, 2022.
[2]. M. Silva et al., "Virtual Assistant for Desktop," International Journal of Innovative Research in Technology (IJIRT), 2022.
[3]. A. Gupta et al., "AI-Powered Virtual Desktop Assistant for Intelligent Automation," International Journal of Innovative Research in Technology, 2023.
[4]. L. T. P. Silva et al., "Desktop Virtual Assistant," International Journal of Research in Advent Technology (IJRASET), 2021.
[5]. S. P. Singh and R. Kaur, "Voice and Text Based Virtual Personal Assistant for Desktop," International Journal of Engineering Applied Sciences and Technology (IJEAST), vol. 5, no. 3, pp. 192–196, 2021
[6]. H. Sharma, "Desktop Assistant: JARVIS, " Zenodo, 2025.
[7]. A. K. Patel, "AI Desktop Assistant using Python [JARVIS]," Zenodo, 2023.
[8]. K. Rajput, "Jarvis Desktop Voice Assistant," GitHub Repository, 2021.
[9]. J. T. Lai et al., "Enhancing Productivity with AI: Personalized Virtual Assistants Using Natural Language Processing," International Journal of Development Research, 2023.
[10]. P. Verma and S. Rao, "The Use of Natural Language Processing in Virtual Assistants and Chatbots," International Journal of Engineering Research & Technology (IJERT), 2023.