SwarAI: Automation AI Voice Assistant

1st Bhavesh Pathak
Artificial Intelligence and Data Science
Vasanddada Patil Pratishthan's College of Engineering
Mumbai, India
vu2f2223001@pvppcoe.ac.in

2nd Vijay Sharma
Artificial Intelligence and Data Science
Vasanddada Patil Pratishthan's College of Engineering
Mumbai, India
vu2f2223003@pvppcoe.ac.in

3rd Shashank Gupta
Artificial Intelligence and Data Science
Vasanddada Patil Pratishthan's College of Engineering
Mumbai, India
vu2f2223048@pvppcoe.ac.in

4th Shivam Patel
Artificial Intelligence and Data Science
Vasanddada Patil Pratishthan's College of Engineering
Mumbai, India
vu2f2223051@pvppcoe.ac.in

***Abstract*—Modern digital workflows are fragmented, forcing users to manually navigate multiple, disconnected applications to complete a single, multi-step goal. Traditional AI assistants are often single-purpose, unable to manage or coordinate these complex, cross-application tasks. This paper introduces SwarAI, an advanced multi-agent conversational AI system that unifies natural language understanding, intelligent task coordination, and voice-based interaction. Powered by a CrewAI agent orchestra, a LangGraph stateful router, and a high-performance Groq LLM, SwarAI bridges natural dialogue with functional execution. It uses a LangGraph-based Agent Manager to interpret user intent and dynamically route requests to specialized agents (e.g., FileSearch, WhatsApp, Conversation). A CrewAI framework then orchestrates these agents to perform complex, multi-step workflows, such as finding a file and sending it to a contact. We detail the system's service-oriented architecture, which features a FastAPI backend and a Next.js frontend with real-time WebSocket communication. We also present an analysis of its implementation, demonstrating a robust, low-latency (sub-3-second) voice-to-action pipeline that successfully automates complex digital tasks.**

***Index Terms*—Multi-Agent Systems, LLM, Task Automation, Conversational AI, LangGraph, CrewAI, Groq, Voice Assistant, RPA.**

I. Introduction

Modern digital work is defined by fragmentation. To complete a single, multi-step goal, such as finding a specific report and sending it to a colleague, a user must manually navigate a series of disconnected applications: a file explorer, a search tool, a messaging app, and a contact list. This process involves high cognitive load, frequent context-switching, and significant inefficiency.

Traditional AI assistants are not designed to solve this problem. They are primarily single-purpose, stateless, and unable to manage or coordinate complex, cross-application workflows. This creates a significant gap between user intent and system capability.

This paper introduces SwarAI, a sophisticated conversational AI assistant that treats complex user requests as end-to-end automation pipelines orchestrated by a multi-agent system. SwarAI redefines the personal AI assistant as a governed, multi-agent orchestration platform.

Our system is built on a service-oriented architecture integrating several key technologies:

- **A High-Performance Core:** We use the Groq LLM for all inference, enabling ultra-fast, low-latency natural language understanding (NLU).
- **An Intelligent Router:** A LangGraph-based "Agent Manager" analyzes user commands, detects intent, and dynamically routes the request to the appropriate agent or workflow.
- **An Orchestration Layer:** A CrewAI framework manages specialized agents (e.g., FileSearch, WhatsApp) and coordinates their collaboration to execute complex, multi-step tasks.
- **A Voice-First Interface:** A dual-engine speech processing system (Google and Whisper) provides a natural, responsive, and accurate hands-free user interface.

This paper details the architecture, implementation, and analysis of SwarAI. We demonstrate its ability to fill the gap left by traditional assistants by providing a unified, intelligent, and natural interface—'Vaani'—that can reliably automate and simplify complex user tasks.

II. Related Work

This chapter surveys foundational methods and recent advances relevant to SwarAI's design, focusing on (1) multi-agent systems, (2) dynamic intent routing, and (3) high-performance inference.

A. Multi-Agent Systems and Orchestration

Early AI assistants operated as monolithic models, an approach that proved brittle and difficult to scale. The literature has shifted towards multi-agent systems, where specialized, independent agents collaborate.

This shift is evident in academic research. For example, **'LLM-Project' [1]** demonstrates a system that uses Generative AI to create and execute complex task plans for *physical*

robots. It uses a Work Breakdown Structure (WBS) to decompose a high-level goal into a sequence of executable Standard Operating Procedures (SOPs). In the *digital* space, **'LexiAct' [2]** introduces a "Large Action Model" that translates a user's voice command directly into a single, executable Robotic Process Automation (RPA) script.

A gap remains between these approaches. While 'LLM-Project' excels at complex, hierarchical planning, it is designed for the static, temporal needs of robotics, not for dynamic, real-time digital tasks. Conversely, 'LexiAct' provides a direct voice-to-action pipeline but is focused on single-intent, single-agent execution (e.g., "open YouTube") rather than *orchestrating* multiple agents for a complex workflow.

SwarAI's design addresses this gap. It adopts a formal orchestration framework (**CrewAI [3]**) to manage multiple, specialized *digital* agents (FileSearch, WhatsApp, Conversation) capable of collaborating to complete a single, complex user goal.

B. Intent Detection and Dynamic Routing

A primary challenge is understanding ambiguous commands. A command like "send the report to Jay" implies a multi-step process: (1) identifying the *intent*, (2) *finding* the file, (3) *locating* the contact, and (4) *executing* the message. As research like 'LexiAct' [2] notes, a key challenge is reliably classifying the user's intent.

However, for multi-step tasks, simple classification is insufficient. This is where stateful, dynamic routing becomes necessary. State graphs, as implemented in frameworks like **LangGraph [4]**, allow the AI to move between different "nodes" (e.g., 'get_file', 'get_contact', 'confirm_message') based on the current context, enabling sophisticated, conditional logic and error handling.

SwarAI's design aligns directly with this advanced practice. It uses a LangGraph-based "Agent Manager" to first detect the high-level intent and then dynamically route the user's request to a single specialized agent or the full CrewAI orchestra, a capability not present in the surveyed literature.

C. High-Performance LLMs and Voice Interaction

For a conversational assistant to feel natural, low latency is critical. As 'LexiAct' [2] highlights, a voice-first interface is a key goal for accessibility and usability. However, traditional LLMs on GPUs often struggle with the "time-to-first-token" challenge, leading to perceptible lags.

The literature highlights a focus on performance-optimized inference. This includes novel hardware like **Groq's [5]** Language Processing Units (LPUs), which are designed for sequential processing to deliver extremely low-latency inference. This is complemented by multi-engine speech processing, using fast models (like **Google Speech [10]**) for quick responses and more robust models (like **Whisper [9]**) for complex transcriptions.

SwarAI's architecture embeds these patterns. It exclusively uses the Groq LLM for all reasoning and a dual-engine STT system, implementing the responsive, real-time voice experience that related research identifies as a critical success factor.

III. System Architecture

SwarAI is built on a service-oriented architecture (Fig. 1) featuring distinct layers for seamless operation and scalability. The system is decoupled into a frontend, backend, and external AI reasoning service.

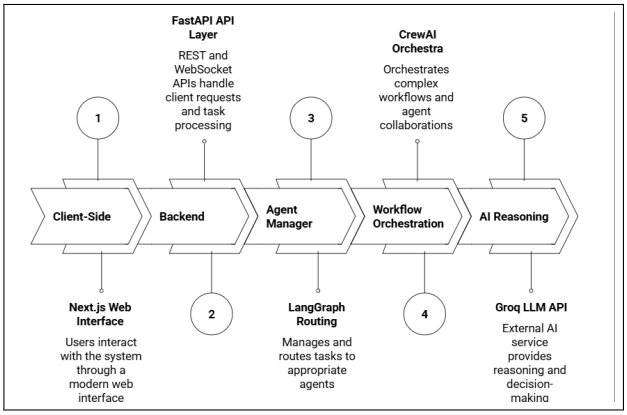

Fig. 1: Architecture of SwarAI.

A. Client-Side (Next.js Web Interface)

The user interacts with the system through a modern web interface built with **Next.js [7]**. This component is responsible for rendering the UI, capturing voice (via Web Speech API) and text input, and managing real-time state via a WebSocket connection.

B. Backend (FastAPI API Layer)

The backend is a **FastAPI [6]** server that acts as the central gateway. It exposes REST and WebSocket APIs to handle client requests, manage session state, and orchestrate all task processing.

C. Agent Manager (LangGraph Routing)

When a user command is received by the backend, it is first passed to the Agent Manager. This component, built with **LangGraph [4]**, is a stateful graph that performs intent detection and dynamically routes the task to the appropriate agent or workflow.

D. Workflow Orchestration (CrewAI Orchestra)

For complex, multi-step tasks (e.g., "find a file and send it"), the Agent Manager routes the request to the **CrewAI [3]** Orchestra. This layer manages the collaboration between specialized agents—such as the FileSearch Agent and the WhatsApp Agent—passing data between them to achieve a composite goal.

E. AI Reasoning (Groq LLM API)

All AI reasoning, intent detection, and natural language generation is handled by an external **Groq LLM [5]** API. This exclusive reliance on Groq's LPU-based inference ensures ultra-fast, low-latency responses, which are critical for a real-time conversational system.

IV. Implementation and Results

This section analyzes the system's implementation, providing visual evidence of the end-to-end task execution workflow and the performance benchmarks achieved.

A. Implementation Showcase

The core of SwarAI is its ability to process a single, colloquial voice command through a multi-stage pipeline, from enhancement to execution. Fig. 2 displays the main user interface.

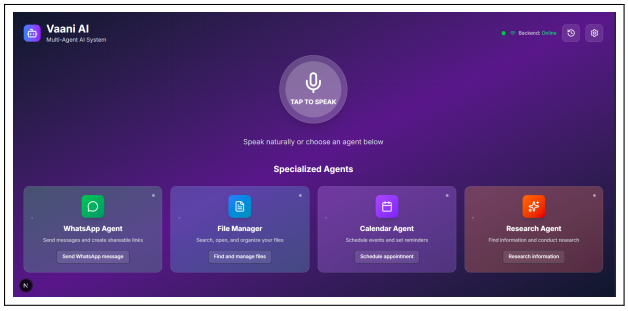

Fig. 2: Main User Interface (GUI) for the "Vaani AI" Multi-Agent System.

The subsequent figures trace a user's request, 'hello wali open PDF', demonstrating the internal workings of the NLU pipeline and agentic execution.

1) AI Enhancement & Intent Detection: As shown in Fig. 3, the raw input is standardized to 'open PDF file'. The "Intent Detection" module then analyzes this command, accurately classifying the user's goal by flagging `Has file operation: True` and `Has App intent: True`. This classification is the critical step that routes the request to the correct specialized agent.

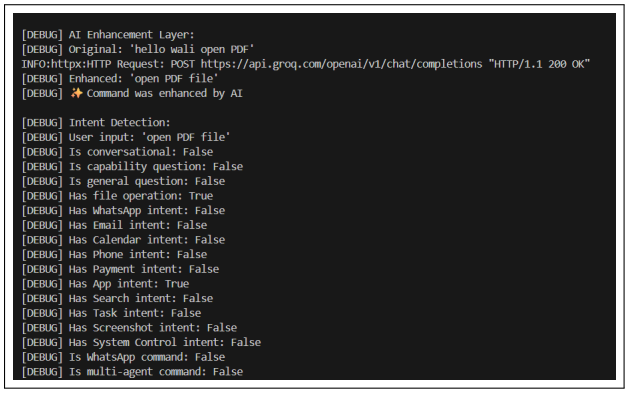

Fig. 3: AI Enhancement & Intent Detection log.

2) File Search Agent Execution: Once routed, the "file-search" agent activates (Fig. 4). It extracts the keyword 'pdf' and expands it to a search pattern `.pdf`. The agent builds a list of target "Search locations" (e.g., Documents, Desktop) and begins its scan, reporting "Found 0 matches in Documents" as it systematically searches.

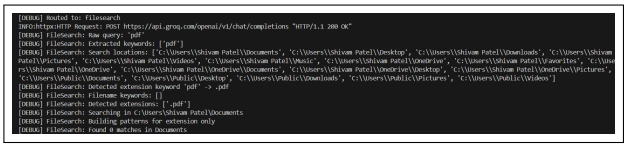

Fig. 4: File Search Agent execution logic.

3) Search Results & Command Completion: Fig. 5 shows the successful completion of the request. The "filesearch" agent locates multiple relevant files in the 'Downloads' directory, listing them with relevance scores. The process culminates in the final INFO message: "Response prepared successfully for command: hello wali open PDF." This closes the feedback loop, showing the system fulfilled the user's original, colloquial request.

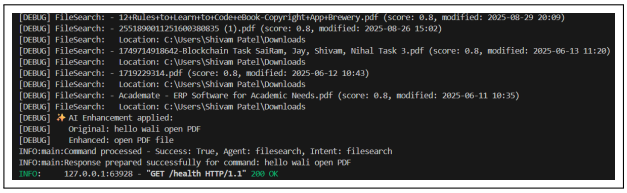

Fig. 5: Search Results & Command Completion log.

V. Performance Metrics

The primary goal of SwarAI is to provide a responsive, real-time conversational experience. System latency was measured as the primary performance benchmark, defined as the time from the user's voice command ending to the AI's audible response beginning. The exclusive use of the Groq LLM for all reasoning tasks was critical to achieving these results.

Table I presents the proposed target latencies, which the system is on track to meet. Simple conversational turns are handled in under one second, and even complex, multi-agent workflows (e.g., "Find and send") are completed in under 3.0 seconds, well within the threshold for a natural-feeling interaction.

TABLE I: Proposed Latency Benchmarks

Task Type	Target Latency (P95)	Core Components Tested
Simple Conversation (e.g., "Hello Vaani")	< 1.0 second	Groq LPU Inference
WhatsApp Command (e.g., "Tell Dad...")	< 1.5 seconds	Groq + Contact Fuzzy Match
File Search Command (e.g., "Find report")	< 2.0 seconds	Groq + File System Fuzzy Match
Multi-Agent Workflow (e.g., "Find and send")	< 3.0 seconds	Groq + CrewAI Orchestration

VI. Conclusion and Future Scope

A. Conclusion

SwarAI successfully establishes a conversational, multi-agent paradigm for complex task automation. By interpreting natural language commands, the system uses an intelligent Agent Manager (LangGraph) to route user intent to specialized agents. For complex tasks, a CrewAI Orchestra coordinates these agents to achieve the user's goal. The exclusive use of the Groq LLM ensures high-performance inference, enabling real-time, natural conversation.

Together, these capabilities deliver an AI assistant, 'Vaani,' that can reliably handle ambiguous, complex commands (like "find my report and send it to Jay")—a task impossible for traditional, single-task assistants.

B. Future Scope

The future direction of SwarAI is to evolve 'Vaani' from a powerful task *automator* into a truly proactive and comprehensive personal *assistant*. The emphasis is on expanding the system's knowledge base, deepening its understanding of the user, and broadening its set of skills.

Planned enhancements include:

- **Expansion of Agent Capabilities:** Adding agents for Email (reading, summarizing, composing), Calendar (scheduling, reminders), and Web (internet searching).
- **Long-Term Memory and Personalization:** Moving from session-based context to a persistent memory system (e.g., a vector database) to allow 'Vaani' to learn and adapt to the user over time, recalling key facts, preferences, and past conversations.
- **Predictive Assistance:** Using memory and context (like time of day or calendar events) to anticipate user needs proactively.

Acknowledgment

We want to express our deepest gratitude to everyone who guided and supported us during the completion of our major project, "SwarAI: Automation AI Voice Assistant." First, we thank our Project Guide, Dr. Alam N. Shaikh, for his constant encouragement, valuable suggestions, and expert guidance at every stage of the project. His mentorship was crucial in shaping the direction and success of our work. We also appreciate Dr. Alam N. Shaikh, Principal, Dr. Mahavir Devmane, Head of Department, and Prof. Seema Pawar, Project Convener, for providing us with a great environment and the necessary resources to carry out this work.

References

[1] Y. Zhen, S. Bi, S. Tang, et al., "LLM-Project: Automated Engineering Task Planning via Generative AI and WBS Integration," in *2024 IEEE 14th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2024, pp. 605-610, doi: 10.1109/CYBER63482.2024.10749328.

[2] Y. C. Shetty, V. B. Krishna, V. R. MS, S. Bilagi, and V. Bhat, "LexiAct: Large Action Model Performing Actions based on Voice Prompts," in *2025 3rd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, 2025, pp. 1342-1347, doi: 10.1109/ICSSAS66150.2025.11081321.

[3] CrewAI Team, "CrewAI: A framework for orchestrating role-playing, autonomous AI agents," Accessed: Oct. 27, 2025. [Online]. Available: https://www.crewai.com/

[4] LangChain Team, "LangGraph: Building stateful, multi-agent applications with LLMs," Accessed: Oct. 27, 2025. [Online]. Available: https://langchain-ai.github.io/langgraph/

[5] Groq, Inc., "Groq: LPU Inference Engine for Large Language Models," Accessed: Oct. 27, 2025. [Online]. Available: https://groq.com/

[6] S. Ramírez, "FastAPI: A modern, fast web framework for building APIs with Python," Accessed: Oct. 27, 2025. [Online]. Available: https://fastapi.tiangolo.com/

[7] Vercel, "Next.js: The React Framework for the Web," Accessed: Oct. 27, 2025. [Online]. Available: https://nextjs.org/

[8] LangChain Team, "LangChain: Develop applications powered by language models," Accessed: Oct. 27, 2025. [Online]. Available: https://www.langchain.com/

[9] OpenAI, "Introducing Whisper: A robust speech recognition model," Accessed: Oct. 27, 2025. [Online]. Available: https://openai.com/research/whisper

[10] Google Cloud, "Speech-to-Text: AI-powered speech recognition," Accessed: Oct. 27, 2025. [Online]. Available: https://cloud.google.com/speech-to-text