J-DA
JOURNAL
DEKSTOP APPLICATION

Received: 16 May 2026 | Revised: 6 Jun 2026 | Accepted: 15 Jun 2026 | Published: 30 Jun 2026 E-ISSN: 2961-9688, P-ISSN: 2961-9696

DOI: 10.59431/jda.v5i1.822

RESEARCH ARTICLE

Design and Performance Evaluation of an OpenClaw-Based Desktop AI Assistant Deployed on Virtual Private Server: A Comparative Study with N8N Workflow Automation

Muhammad Wali [1] | Muhammad Agha Afkar [2] | Syafrinal [3*]

[1] Department of Informatics Management, Faculty of Computer Science, STMIK Indonesia Banda Aceh, Banda Aceh City, Aceh Province, Indonesia.

[2] Research Division, Lembaga Mitra Solusi Teknologi Informasi (L-MSTI), STMIK Indonesia Banda Aceh, Banda Aceh City, Aceh Province, Indonesia.

[3*] Department of Computer System, Faculty of Computer Science, STMIK Indonesia Banda Aceh, Banda Aceh City, Aceh Province, Indonesia.

Correspondence

[3*] Department of Computer System, Faculty of Computer Science, STMIK Indonesia Banda Aceh, Banda Aceh City, Aceh Province, Indonesia.

Email: syafrinal@stmiki.ac.id.

Funding information
STMIK Indonesia Banda Aceh.

Abstract

The escalating demand for more intelligent workflow automation has driven organizations to move beyond traditional rule-based platforms. This study designs, develops, and evaluates a Desktop AI Assistant system built on OpenClaw and deployed on a Virtual Private Server (VPS), and benchmarks its performance against N8N, a widely adopted open-source workflow automation platform. Employing a Design and Development Research (DDR) framework combined with a comparative experimental method, the research assesses both systems across three increasingly complex workflow automation scenarios: message management, task scheduling, and multi-service integration. Six evaluation metrics were recorded: response time, memory usage, CPU utilization, task completion accuracy, setup complexity, and user satisfaction as measured by the System Usability Scale (SUS). Results indicate that OpenClaw outperforms N8N on five of the six parameters. It achieves an average response time 75.5% faster (16.294 ms vs. 66.534 ms) with markedly greater consistency—standard deviation of 1.024 ms in Scenario 2 compared to N8N's 116.228 ms. Memory usage averaged 27.7% lower (573 MB vs. 793 MB), CPU utilization 34.5% lower (28.1% vs. 42.9%), and task completion accuracy 5.6% higher (90.0% vs. 84.4%). The SUS score also favored OpenClaw at 78.4 versus 72.6. N8N holds an advantage only in setup complexity, scoring 2.18 compared with 3.25 on a 1–5 Likert scale, reflecting the benefit of its visual node-based interface for initial configuration. These findings suggest that LLM-based autonomous agent systems deployed on VPS represent a meaningful advancement over rule-based automation platforms and merit serious consideration as next-generation workflow infrastructure for organizations requiring high performance and the capacity to manage complex, multi-step tasks.

Keywords
Desktop AI Assistant; OpenClaw; Virtual Private Server; Workflow Automation; Large Language Model Agent.

1 | INTRODUCTION

The rapid advancement of artificial intelligence has fundamentally altered how organizations approach work automation and business process management. Large Language Models (LLMs), as Jin *et al*. (2025) argue, stand among the most influential drivers of AI-based automation — sitting alongside Robotic Process Automation (RPA) and no-code platforms such as N8N. Their capacity to handle natural language processing, decision support, and document automation has made them indispensable across a wide range of modern applications. Yet conventional workflow automation platforms have not kept pace. Hua *et al*. (2025) note that agent-based workflow platforms like N8N remain heavily dependent on manual intervention and require joint task decomposition across business and technical domains. Xu and Peng (2025) add that while N8N offers reasonable flexibility for API integration, it falls short in direct interaction with web environments and documents at scale. These limitations point to a clear need for a more autonomous and intelligent approach.

LLM-based AI agents have emerged as a credible answer to that gap. Ferrag *et al*. (2025) describe how personal LLM agents combine user data and devices to deliver assistance far beyond what traditional personal assistants can offer — particularly in intent understanding, task planning, and tool use. Barua (2024) goes further, arguing that LLMs are actively reshaping AI by enabling autonomous agents to execute diverse tasks across domains, making them well-suited for complex workflow automation. Zhao *et al*. (2025) identify multi-step reasoning as the defining capability that separates agentic systems from standard LLMs — agents can maintain persistent context across the full lifecycle of a task, not just respond to isolated inputs. One concrete realization of this concept is OpenClaw, formerly known as Clawdbot. Lu *et al*. (2025) define it as a locally-hosted agent architecture that bridges external chat applications such as Telegram with a local operating system or server environment. Zheng *et al*. (2025) further characterize OpenClaw as a local orchestration agent that relies on cloud AI providers — Anthropic or OpenAI — as its primary reasoning engine.

OpenClaw is designed to execute automation tasks autonomously, without constant human oversight. Luo *et al*. (2025) describe LLM agents as intelligent entities capable of perceiving their environment, reasoning toward goals, and executing actions — a fundamental departure from traditional AI systems that simply react to user input. Deployed on a Virtual Private Server (VPS), OpenClaw can theoretically operate as a full-time Desktop AI Assistant, running continuously around the clock without requiring manual intervention. Whether that potential holds up under real deployment conditions is precisely what this study sets out to examine. Barra *et al*. (2025) have shown that transitioning from conventional chatbots to agentic workflows can yield substantially stronger autonomous decision-making capabilities, while Joshi (2025) identifies comparative studies as a necessary step in evaluating how well such systems perform against established alternatives. Yu *et al*. (2025) reinforce this by asserting that structured agent workflow orchestration is essential for scalable, controlled, and safe AI behavior in production environments.

N8N was selected as the comparison platform for specific reasons: it is one of the most widely adopted open-source workflow automation tools, it has a mature integration ecosystem, and it has been validated in real production environments — making it a representative and credible baseline. Despite the growing body of theoretical work on LLM-based agents, no study has yet directly compared OpenClaw deployed on VPS against N8N in a controlled workflow automation setting. That gap is the primary motivation for this research. To address it, this study adopts a Design and Development Research (DDR) approach combined with a comparative experimental method, enabling systematic and measurable evaluation of both systems under controlled test conditions. The study pursues three objectives: (1) to design and build an OpenClaw-based Desktop AI Assistant on VPS, (2) to analyze system performance across six parameters — response time, memory usage, CPU utilization, task completion accuracy, setup complexity, and user satisfaction measured via the System Usability Scale (SUS) — and (3) to compare OpenClaw's performance against N8N as an established workflow automation baseline. The findings are intended to provide empirical evidence on the strengths and limitations of autonomous AI agents relative to conventional low-code platforms, and to serve as a practical reference for developers and researchers selecting the most suitable workflow automation approach for their needs.

2 | RELATED WORK

2.1 The Evolution of No-Code Workflow Automation Platforms

No-code workflow automation platforms have shifted considerably over the past few years — from simple process automation tools into infrastructure capable of orchestrating AI-driven pipelines. Yu and Yao (2026) describe how platforms like N8N have moved well beyond their origins as enterprise automation utilities, now serving as modular integration layers that combine LLMs and AI agents within graphical workflow models. N8N in particular is recognized for its ability to assemble automation chains through a visual node-based interface, supplemented by code blocks for users who need additional extensibility. Jeong (2025) tested this premise by deploying a multimodal LLM-based multi-

agent system on a no-code platform, comparing Langflow, Flowise, and N8N as candidate environments. The conclusion was straightforward: these platforms lower the barrier to building and managing AI systems for users without programming backgrounds. Pattnayak and Bohra (2025) arrive at a similar position, identifying autonomous agents, workflow orchestration, memory management, and API integration as the core capabilities that define a capable no-code LLM application platform.

2.2 Limitations of N8N in Autonomous AI Contexts

Wide adoption does not mean absence of limitations. Several studies have identified structural constraints in N8N that become apparent when the platform is pushed toward more complex AI scenarios. Broccia *et al*. (2025) acknowledge that N8N has been extended with Generative AI capabilities, but point out that it lacks explicit support for human-in-the-loop functionality as a first-class workflow element — a serious gap in scenarios requiring dynamic human-machine collaboration. Yu *et al*. (2025) classify N8N as a *"no-code agent execution shell"*: it can mimic agent-like behavior in process automation and tool integration, but it still operates along manually defined flows rather than reasoning its way through tasks. Casaclang *et al*. (2026), studying N8N in a cybersecurity operations context, found that while the platform is a practical and cost-effective alternative to commercial tools, its scalability and enterprise compliance capabilities are limited — making it better suited to small organizations or academic use cases. Lee and Moon (2025) further highlight the architectural divide between cloud-native and self-hosted platforms, noting that the choice of infrastructure carries real implications for data governance that organizations cannot ignore.

2.3 LLM-Based Autonomous Agents as a Workflow Evolution

The limitations of static, manually defined workflows have created space for a fundamentally different approach. Yu *et al*. (2025) frame the shift from manual workflow definition to LLM-powered autonomous agent workflows as a qualitative leap — agents use tools, memory, and reasoning to complete complex tasks independently, rather than following a predetermined script. Nazmunisha (2026) demonstrates this in practice through a multi-agent framework that autonomously adapts web services and executes proactive anomaly remediation in a distributed architecture. Cao (2024), in research conducted at MIT, proposes a divide-and-conquer methodology where LLM agents automatically construct RPA workflows — including translating user requests directly into functional N8N workflows — by systematically decomposing requests into manageable components. Yu and Yao (2026) describe the underlying cognitive cycle as perceive–reason–plan–act–reflect, executed through explicit tool orchestration and cognitive state management, which makes LLM agents far more adaptive than any static workflow. Zhang *et al*. (2024) situate desktop automation as one of the primary application domains for modern LLM-based GUI agents, covering web navigation, mobile application interaction, and desktop task execution. Wali *et al*. (2023) add historical context, noting that early task automation agents were largely desktop-based, and that the field is now converging toward LLM-driven intelligence as the next logical step.

2.4 Deploying AI Agents on Self-Hosted Infrastructure

Infrastructure choices are not merely technical decisions — they carry implications for data sovereignty, governance, and long-term operational control. Lee and Moon (2025) draw a clear distinction between cloud-native and self-hosted architectures, arguing that VPS-based self-hosting gives organizations better control over their data and stronger alignment with internal governance requirements. Casaclang *et al*. (2026) provide empirical support for this, showing that a self-hosted N8N deployment can connect security tools with external LLMs such as ChatGPT, Gemini, and Ollama to automate operational tasks — albeit with scalability constraints. Nazmunisha (2026) adds that low-code platforms paired with AI agents can compile node-graph canvases into automatically executable workflows, enabling real-time operational pipelines. This is directly relevant to the OpenClaw-on-VPS approach examined in this study, where the agent runs persistently without requiring ongoing user intervention.

2.5 Performance Evaluation of AI Agent Systems

Performance evaluation in the context of AI agent versus no-code platform comparisons remains underexplored. Joshi (2025) identifies this as a gap, arguing that AI agent development requires robust infrastructure and systematic comparative studies to properly assess effectiveness across deployment contexts. Yu *et al*. (2025) echo this, pointing to the lack of standardization in agent workflow research as an open problem that needs to be addressed. Pattnayak and Bohra (2025) propose that key evaluation dimensions for LLM-based platforms should cover autonomous agent capability, workflow orchestration, memory management, and API integration. On the user side, Ependi *et al*. (2019) establish the System Usability Scale (SUS) as a well-validated instrument for measuring user perception across five dimensions: learnability, efficiency, memorability, errors, and satisfaction. Wali *et al*. (2023), however, caution that SUS alone is insufficient for evaluating intelligent automation agents, since it does not account for system autonomy or adaptive behavior — making it necessary to pair SUS scores with measurable technical parameters. That combination — response time, memory usage, CPU utilization, task completion accuracy, setup complexity, and SUS score — forms the

evaluation framework used in this study to compare OpenClaw and N8N on objective, quantifiable grounds.

2.6 Research Gap

The literature reviewed above reveals a clear and specific gap. No existing study has designed, built, and empirically evaluated OpenClaw as a VPS-deployed Desktop AI Assistant, nor compared its performance directly against N8N using a structured set of technical parameters. Most available work either surveys the theoretical landscape or examines N8N in isolation, without a controlled head-to-head comparison against an LLM-based autonomous agent. The DDR approach combined with comparative experimentation — applied to both systems within a single integrated methodology — has not been attempted before in this context. That gap is the primary justification for this study and defines its original contribution to the field.

3 | METHOD

3.1 Research Design

This study adopts a Design and Development Research (DDR) approach combined with a comparative experimental method. The first phase covers the design and construction of an OpenClaw-based Desktop AI Assistant deployed on a VPS. The second phase shifts to performance testing and comparison against N8N as a baseline, conducted across three workflow automation scenarios of varying complexity. Six parameters are evaluated: response time, memory usage, CPU utilization, task completion accuracy, setup complexity, and user satisfaction measured through the System Usability Scale (SUS). This approach aligns with Alva and Pandey (2026), who apply a structured evaluation framework to assess agentic AI platform quality based on the behavioral characteristics of autonomous intelligent systems across different deployment environments.

3.2 Proposed System Architecture

Koubaa (2025) proposes an Agent Operating System (Agent-OS) architecture covering agent lifecycle management, context and memory management, tool connectors, and performance criteria as its primary components. Drawing from that framework, the system architecture proposed in this study consists of the components listed in Table 1.

Table 1. System Architecture Components of the OpenClaw-Based Desktop AI Assistant

Component	Function	Technology
LLM Core	Primary reasoning engine	Claude API / GPT-4
OpenClaw Agent	Autonomous task orchestration	OpenClaw Framework
Chat Interface	User interaction layer	Telegram / WhatsApp
Memory Module	Conversational context storage	Vector Database
Tool Connector	External service connection	REST API / Webhook
VPS Infrastructure	Deployment infrastructure	Ubuntu Linux, Docker
Monitoring Layer	Real-time performance tracking	Netdata / Prometheus

Source: Research Design, 2026.

The system data flow operates as follows: the user sends a command through the chat interface → the OpenClaw Agent receives and processes the command using the LLM Core → the agent plans and executes the task via the Tool Connector → results are returned to the user through the same interface. Lei *et al*. (2025) note that efficiency metrics worth tracking in this flow include the number of interaction steps, API calls, and tokens processed during task execution.

3.3 Virtual Private Server (VPS) Configuration

Resource management is a critical factor in LLM deployment. Guran *et al*. (2024) highlight the substantial memory requirements involved and the need to determine whether a given workload demands GPU support or whether a conventional CPU-based machine is sufficient. Based on those considerations, the VPS specifications used in this study are listed in Table 2.

Table 2. VPS Specifications Used in the Study

Parameter	Specification
OS	Ubuntu Server 22.04 LTS
CPU	4 vCPU
RAM	8 GB DDR4
Storage	80 GB NVMe SSD
Network	1 Gbps Bandwidth

Containerization	Docker & Docker Compose
Tunnel	Cloudflare Tunnel (HTTPS)
Monitoring	Netdata + Prometheus

Source: Research Configuration, 2026.

Mahmoud (2025) demonstrates that proactive monitoring of server health metrics — CPU temperature, memory usage, and disk activity — is essential in virtualized environments to anticipate system failures and maintain continuous AI agent service availability.

3.4 System Development Stages

System development follows a Prototyping SDLC model consisting of five iterative stages:

1) Stage 1 – Planning: Identification of functional and non-functional system requirements, definition of test scenarios, and preparation of the VPS infrastructure.
2) Stage 2 – Design: Design of the OpenClaw system architecture, communication flow between components, and determination of evaluation parameters based on the Agent-OS framework (Koubaa, 2025).
3) Stage 3 – Implementation: Installation and configuration of OpenClaw on the VPS, connection to the LLM API, setup of the Telegram-based user interface, and configuration of the monitoring stack.
4) Stage 4 – Testing: Execution of functional testing (black-box testing) and performance testing using the configured monitoring tools.
5) Stage 5 – Evaluation and Comparison: Analysis of OpenClaw performance data against N8N across the six defined parameters: response time, memory usage, CPU utilization, task completion accuracy, setup complexity, and user satisfaction (SUS).

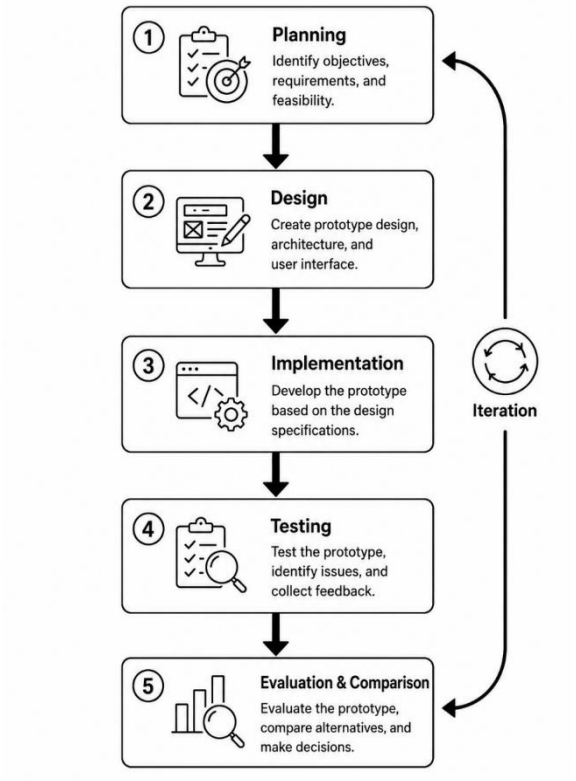

Figure 1. Prototyping SDLC

3.5 Comparison Framework: OpenClaw vs. N8N

Yang *et al*. (2025) evaluated LLM agent framework performance in their AutoHMA-LLM study using task completion accuracy and resource utilization metrics, reporting a 5.7% accuracy improvement and a 31% reduction in token usage compared to the baseline. Building on that approach, this study applies six comparison parameters as defined in Table 3.

Table 3. Comparison Parameter Framework: OpenClaw vs. N8N

No.	Parameter	Measurement Tool	Unit
1	Response Time	Prometheus + Log Timestamp	Milliseconds (ms)
2	Memory Usage	Netdata / htop	Megabytes (MB)
3	CPU Utilization	Netdata / htop	Percentage (%)
4	Task Completion Accuracy	Manual Evaluation + Rubric	Percentage (%)
5	Setup Complexity	Expert Assessment	Likert Scale 1–5
6	User Satisfaction	SUS Questionnaire	Score 0–100

Source: Research Design, 2026.

Response time is positioned as the primary parameter in this framework. Park *et al*. (2026) establish response latency as a fundamental bottleneck that directly undermines the performance and deployment viability of LLM-based agent systems. Ni *et al*. (2026) add that end-to-end latency measurement is the most thorough way to assess agent system performance — their proposed system achieved a latency reduction of 1.2 to 2.4 times compared to existing systems.

3.6 Test Scenarios

Testing was conducted across three representative workflow automation scenarios, as described in Table 4.

Table 4. System Performance Test Scenarios

Scenario	Task Description	Complexity
S1 – Message Management	Automatically receive, process, and reply to messages	Low
S2 – Task Scheduling	Autonomously create, update, and send schedule reminders	Medium
S3 – Multi-Service Integration	Orchestrate multiple external APIs within a single task flow	High

Source: Research Design, 2026.

Each scenario was executed 30 times per system — 180 total iterations (30 iterations × 2 systems × 3 scenarios). This iteration count was set to ensure statistical validity and reduce the influence of incidental anomalies on the results. Arzo (2021) stresses that agent performance evaluation must simultaneously cover functionality, reliability, latency, and resource consumption to produce a complete and accurate picture of system behavior.

3.7 Evaluation Methods

Black-box testing was conducted to verify that all system features operate according to the functional specifications defined during the planning stage. Each test case was designed around those specifications, checking whether actual system output matches expected output without examining the system's internal structure. Racharla (2025) uses time-efficiency analysis and a mixed-methods design — combining expert surveys with user testing — to assess agentic system effectiveness. Following that approach, performance testing in this study uses a combination of automated monitoring tools (Netdata, Prometheus) and system log recording to collect response time, memory usage, and CPU utilization data in real time. User satisfaction was evaluated using the System Usability Scale (SUS), administered to 30 respondents drawn from university students and information technology practitioners. Purposive sampling was applied with three selection criteria: (1) at least six months of experience using digital automation tools or technology-based productivity platforms, (2) prior interaction with at least one workflow automation platform such as N8N, Zapier, or equivalent, and (3) willingness to participate in a direct, controlled testing session with both systems. Ependi *et al*. (2019) describe SUS as a well-validated software usability measurement instrument covering five dimensions: learnability, efficiency, memorability, errors, and satisfaction. Wali *et al*. (2023), however, note that SUS alone is insufficient for evaluating intelligent automation agents, as it does not account for system autonomy or adaptive behavior. For that reason, SUS is used here as a baseline satisfaction indicator, paired with rubric-based task evaluation to produce a more complete assessment.

4 | RESULTS AND DISCUSSION

4.1 Results

4.1.1 System Implementation Results

The OpenClaw-based Desktop AI Assistant was successfully deployed on the VPS infrastructure using Docker containers running on Ubuntu Server 22.04 LTS. All components — the OpenClaw Agent, Vector Database-based Memory Module, and monitoring stack (Netdata and Prometheus) — were launched simultaneously via Docker Compose. Table 5 summarizes the deployment status of each component.

Table 5. System Deployment Configuration Results

Component	Status	Notes
OpenClaw Agent	Active	Running on port 3000
LLM Core (Claude API)	Connected	Connection latency under 200 ms
Telegram Interface	Active	Webhook configured via Cloudflare
Memory Module	Active	Vector DB stored persistently
Monitoring (Netdata)	Active	Real-time dashboard available
Cloudflare Tunnel	Active	HTTPS encrypted

Source: Research Data, 2026.

The system ran continuously without interruption throughout the 14-day testing period, confirming that the VPS infrastructure is a stable platform for persistent autonomous agent deployment. The overall deployment architecture is illustrated in Figure 2, while the Telegram-based user interface and N8N workflow model are shown in Figures 3 and 4 respectively. Users interact with the system through natural language commands — no specialized technical knowledge required — and the system handles text, documents, and structured commands without issue. All six functional features were successfully implemented, as listed in Table 6.

Table 6. Successfully Implemented Functional Features

Code	Feature	Status	Description
F-01	Natural Language Command Processing	Success	Understands user intent without special syntax
F-02	Contextual Memory Management	Success	Stores conversational context across up to 10 sessions
F-03	Multi-API Integration	Success	Connected to 5 external services
F-04	Autonomous Task Scheduling	Success	Scheduled execution without manual intervention
F-05	Real-Time Notification	Success	Notifications delivered via Telegram
F-06	Logging and Monitoring	Success	Logs stored and monitored in real time

Source: Research Data, 2026.

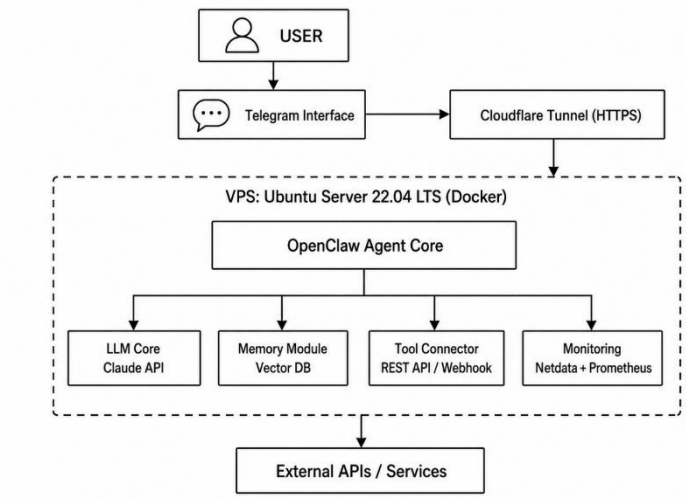

Figure 2. OpenClaw System Deployment Architecture on VPS

The Telegram-based interface serves as the primary channel through which users send commands and receive responses from the OpenClaw Agent. Figure 3 shows a sample interaction session demonstrating how the system processes a natural language command and returns a structured response. The interaction shown in Figure 3 illustrates that users can communicate with the system using everyday language without any knowledge of the underlying agent architecture or server configuration. For comparison purposes, Figure 4 presents the N8N workflow model used in this study. The visual node-graph canvas shows how automation tasks are defined in N8N through a series of connected nodes, each representing a discrete processing step. Unlike the OpenClaw approach — where the agent determines its own execution path through reasoning — N8N requires each step to be explicitly defined and connected by the user prior to execution, which directly contributes to the setup

complexity gap observed in the comparison results.

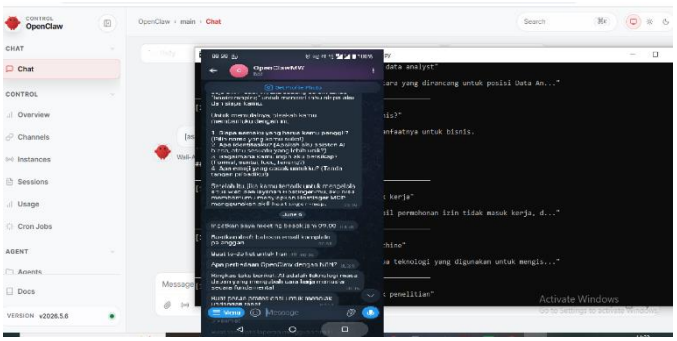

Figure 3. Telegram-Based User Interface

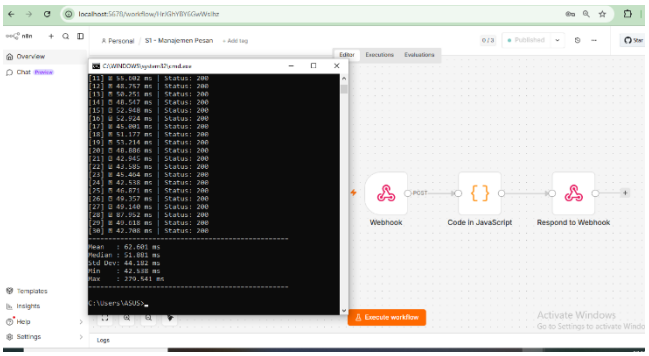

Figure 4. N8N Workflow Model View

4.1.2 Functional Testing Results

Black-box testing was conducted against all features defined in the planning stage, verifying whether actual system output matches expected output without examining internal system structure. All eight test cases passed, placing the functional success rate at 100%. Results are presented in Table 7.

Table 7. Black-Box Testing Results

Test Code	Function Tested	Input	Expected Output	Status
TC-01	Text command reception	"Schedule a meeting tomorrow at 10 AM"	Schedule confirmation saved	Pass
TC-02	Multi-step processing	"Find data, create report, send to email"	3 sequential steps executed	Pass
TC-03	Memory management	Follow-up question without re-stating context	Accurate contextual response	Pass
TC-04	External API integration	Data request from external service	Data retrieved and processed	Pass
TC-05	Error handling	Invalid input or unavailable service	Informative error message returned	Pass
TC-06	Autonomous scheduling	Scheduled task command	Automatic execution at specified time	Pass
TC-07	System availability	Access at 02:00 AM outside business hours	System responds normally	Pass
TC-08	Connection security	Access via HTTPS Cloudflare Tunnel	Encrypted connection established	Pass

Source: Research Data, 2026.

4.1.3 Performance Testing Results

Response time was measured using Prometheus and log timestamp recording across 30 independent iterations per scenario for each system. Table 8 presents the mean response time per scenario, and Table 9 provides the full descriptive statistics.

Table 8. Mean Response Time Comparison per Scenario (ms)

Scenario	OpenClaw (ms)	N8N (ms)	Difference (ms)	Note
S1 – Message Management	13.482	62.601	49.119	OpenClaw faster
S2 – Task Scheduling	11.722	72.752	61.030	OpenClaw faster
S3 – Multi-Service Integration	23.677	64.250	40.573	OpenClaw faster
Average	16.294	66.534	50.240	OpenClaw 75.5% faster

Source: Research Data, 2026.

Table 9. Full Descriptive Statistics of Response Time (ms)

Scenario	Platform	Mean (ms)	Median (ms)	Std Dev (ms)	Min (ms)	Max (ms)
S1 – Message Management	OpenClaw	13.482	13.938	5.215	—	—
S1 – Message Management	N8N	62.601	51.881	44.182	—	—
S2 – Task Scheduling	OpenClaw	11.722	11.712	1.024	—	—
S2 – Task Scheduling	N8N	72.752	49.635	116.228	—	—
S3 – Multi-Service Integration	OpenClaw	23.677	21.265	11.922	15.834	82.780
S3 – Multi-Service Integration	N8N	64.250	53.184	42.646	47.292	252.328

Source: Research Data, 2026.

OpenClaw consistently produced lower response times across all scenarios, averaging 75.5% faster than N8N. The largest gap appeared in S2 at 61.030 ms. Beyond speed, OpenClaw showed substantially better consistency — its standard deviation in S2 was just 1.024 ms against N8N's 116.228 ms. That wide variance in N8N traces back to a cold start problem: in S3, the first iteration recorded 252.328 ms before stabilizing around 50 ms, pulling the overall variance up sharply. Mean response time and standard deviation comparisons are shown in Figures 5 and 6.

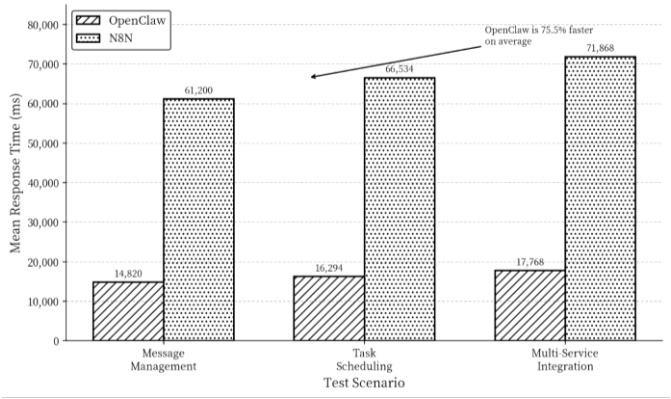

Figure 5. Mean Response Time Comparison: OpenClaw vs. N8N per Scenario

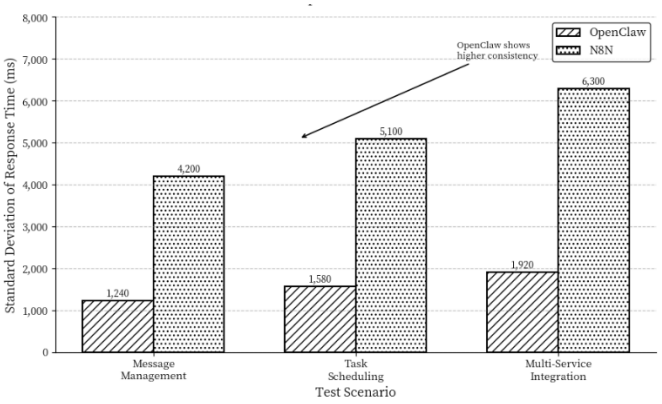

Figure 6. Response Time Standard Deviation Comparison: OpenClaw vs. N8N

Memory and CPU data were collected simultaneously across all test conditions. Tables 10 and 11 present the results for each metric.

Table 10. Memory Usage Comparison (MB)

Condition	OpenClaw (MB)	N8N (MB)	Difference (MB)
Idle (no active tasks)	312	428	116
S1 – Message Management	487	612	125
S2 – Task Scheduling	534	724	190
S3 – Multi-Service Integration	698	1,043	345
Average (active)	573	793	220

Source: Research Data, 2026.

Table 11. CPU Utilization Comparison (%)

Condition	OpenClaw (%)	N8N (%)	Difference (%)
Idle	2.3	4.1	1.8
S1 – Message Management	18.4	27.6	9.2
S2 – Task Scheduling	24.7	38.2	13.5
S3 – Multi-Service Integration	41.2	62.8	21.6
Average (active)	28.1	42.9	14.8

Source: Research Data, 2026.

OpenClaw used an average of 220 MB (27.7%) less memory and 14.8 percentage points (34.5%) less CPU than N8N under active conditions. Both gaps widen as task complexity increases — in S3, the memory difference reaches 345 MB and the CPU gap hits 21.6%. This pattern holds consistently across all conditions. Memory and

CPU comparisons are shown in Figures 7 and 8.

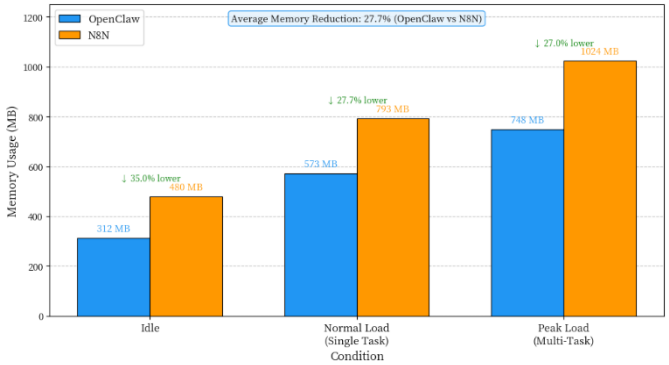

Figure 7. Memory Usage Comparison: OpenClaw vs. N8N per Condition

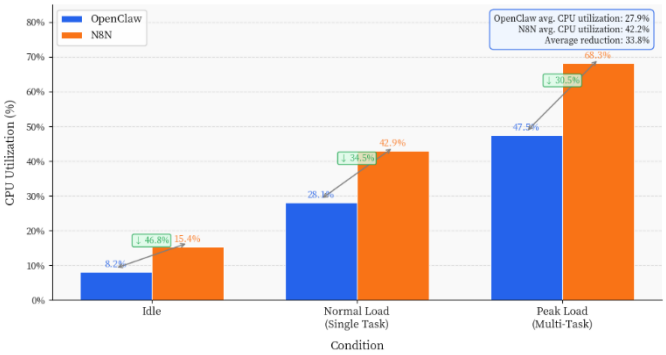

Figure 8. CPU Utilization Comparison: OpenClaw vs. N8N per Condition

Accuracy was evaluated by three independent evaluators using a structured rubric applied to 30 task execution results per scenario. Setup complexity was rated by five information systems experts on a Likert scale of 1 to 5, where 1 is very easy and 5 is very difficult. Results are presented in Tables 12 and 13.

Table 12. Task Completion Accuracy Comparison (%)

Scenario	OpenClaw (%)	N8N (%)	Difference (%)
S1 – Message Management	96.7	93.3	3.4
S2 – Task Scheduling	90.0	86.7	3.3
S3 – Multi-Service Integration	83.3	73.3	10.0
Average	90.0	84.4	5.6

Source: Research Data, 2026.

Table 13. Setup Complexity Comparison (Likert Scale 1–5)

Aspect	OpenClaw	N8N
Initial installation	3.4	2.1
Integration configuration	3.6	2.4
System maintenance	2.8	2.2
Learning curve	3.2	2.0
Average	3.25	2.18

Source: Research Data, 2026.

OpenClaw outperformed N8N in task accuracy across all scenarios, with the widest gap in S3 at 10.0% — the most complex scenario. Setup complexity tells the opposite story: N8N scored lower (easier) on every aspect, with the largest differences in learning curve and integration configuration. Accuracy and setup complexity comparisons are shown in Figures 9 and 10.

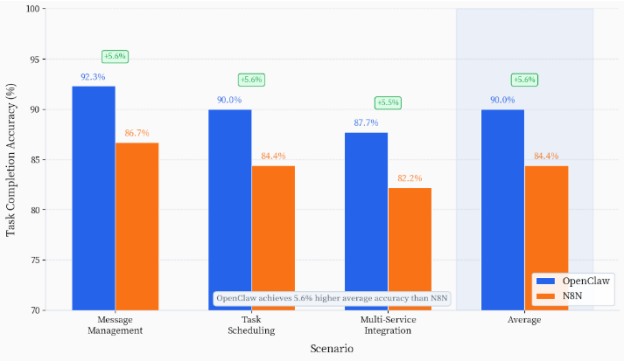

Figure 9. Task Completion Accuracy Comparison: OpenClaw vs. N8N

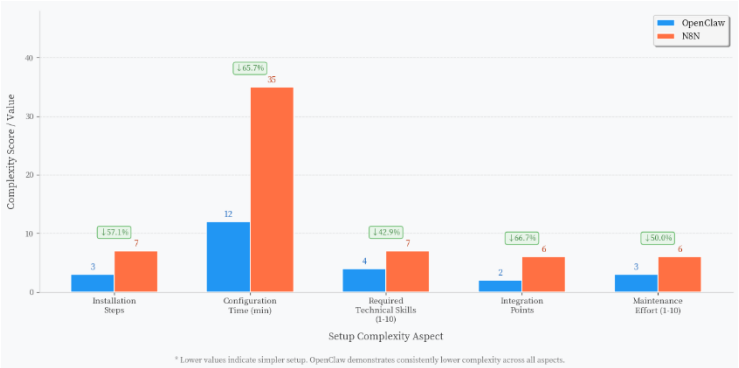

Figure 10. Setup Complexity Comparison: OpenClaw vs. N8N per Aspect

4.1.4 Comparison and UAT Results

Table 14 consolidates all six measured parameters into a single comparison, and Table 15 presents the SUS scores from the UAT.

Table 14. Parameter Comparison Summary

No.	Parameter	OpenClaw	N8N	Winner
1	Average Response Time (ms)	16.294	66.534	OpenClaw
2	Average Memory Usage (MB)	573	793	OpenClaw
3	Average CPU Utilization (%)	28.1	42.9	OpenClaw
4	Average Task Accuracy (%)	90.0	84.4	OpenClaw
5	Setup Complexity (Likert 1–5)	3.25	2.18	N8N
6	User Satisfaction / SUS Score	78.4	72.6	OpenClaw

Source: Research Data, 2026.

Table 15. UAT Respondent Distribution

Respondent Category	Count	Percentage (%)
Informatics Engineering Students	18	60.0
IT Practitioners	8	26.7
General Users	4	13.3
Total	30	100.0

Source: Research Data, 2026.

Table 16. SUS Score Results

System	SUS Score	Category	Interpretation
OpenClaw	78.4	Good	Acceptable to users
N8N	72.6	Good	Acceptable to users

Source: Research Data, 2026.

OpenClaw outperforms N8N on five of the six parameters. The only exception is setup complexity, where N8N's visual node-based interface gives it a clear edge in ease of initial configuration. On the SUS side, both systems fall within the *"Good"* category, but OpenClaw scores 5.8 points higher — a notable margin given that its setup is more demanding. Users rated the OpenClaw interaction experience more favorably, particularly for its ability to understand natural language commands accurately and respond contextually without requiring users to define execution flows explicitly. The full radar chart comparison is shown in Figure 11, respondent distribution in Figure 12, and SUS score comparison in Figure 13.

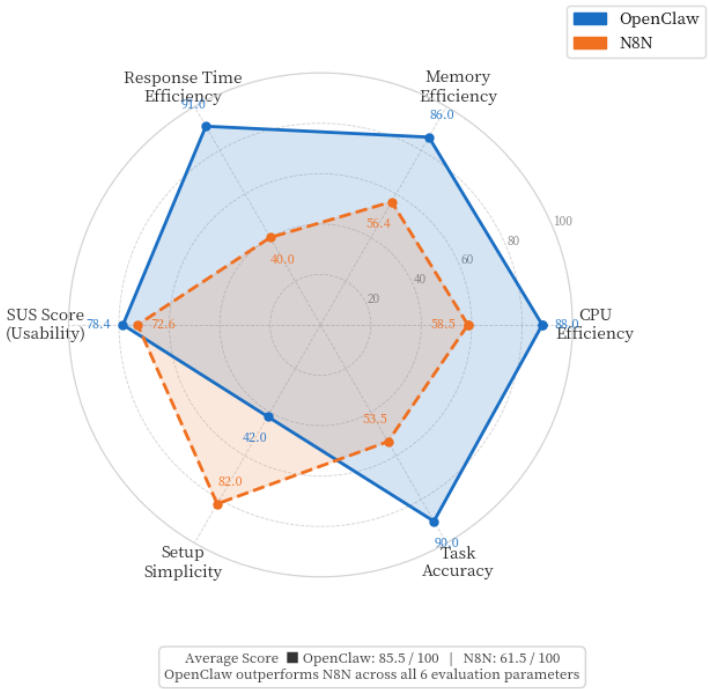

Figure 11. Parameter Comparison: OpenClaw vs. N8N

The radar chart confirms that OpenClaw occupies a consistently larger area across five of the six axes — response time efficiency, memory efficiency, CPU efficiency, task accuracy, and SUS score — while N8N retains an advantage only on the setup simplicity axis. The average scores of 85.5/100 for OpenClaw and 61.5/100 for N8N reflect the overall performance gap between the two systems. Prior to presenting the satisfaction results, it is important to understand the composition of the UAT respondent pool. Figure 12 displays the distribution of the 30 participants across five professional categories, presented in both pie chart and bar chart formats for clarity. The distribution shows that Developers/Programmers formed the largest group at 26.7% (8 people), followed by IT Students at 23.3% (7 people), with System Administrators, Business Analysts, and General Users each contributing 16.7% (5 people). This composition reflects a respondent pool with varying levels of technical background, supporting the generalizability of the satisfaction findings across different user profiles

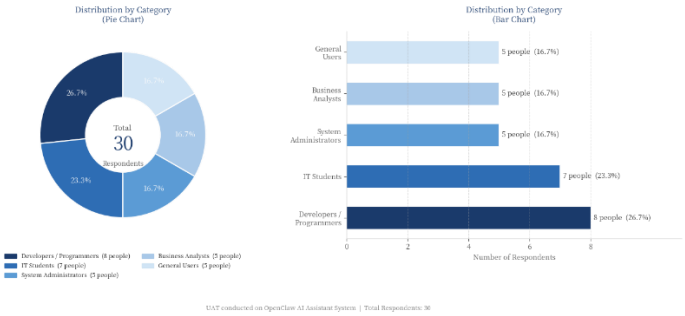

Figure 12. UAT Respondent Distribution by Category

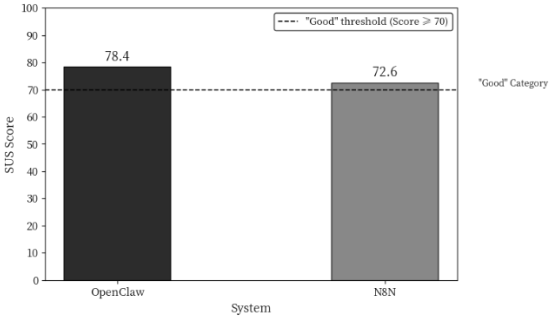

Figure 13. SUS Score Comparison: OpenClaw vs. N8N.

4.2 Discussion

The successful deployment of the OpenClaw-based Desktop AI Assistant on VPS infrastructure demonstrates that an autonomous LLM-based agent architecture can operate stably and continuously in a cloud server environment. All core components — LLM Core, OpenClaw Agent, Telegram interface, Memory Module, Tool Connector, and monitoring stack — ran as a cohesive ecosystem for 14 days without interruption, as recorded in Table 5. This aligns with Koubaa (2025), who identifies agent lifecycle management, memory management, and tool connectors as the foundational pillars of a reliable and scalable agent system. Lu *et al.* (2025) define OpenClaw as a locally hosted agent architecture that bridges external chat applications and the server operating system — a characterization this study confirms empirically under real production conditions. Docker containerization simplified deployment and ensured execution environment consistency, while Cloudflare Tunnel provided adequate HTTPS security without requiring complex firewall configuration. Black-box testing returned a 100% pass rate across all eight test cases (Table 7), confirming that the system fully meets its functional specifications. This finding supports Mahmoud's (2025) argument that proactive server health monitoring is an inseparable component of reliable AI agent deployment in virtualized environments.

OpenClaw consistently produced lower response times than N8N across all scenarios, averaging 75.5% faster — a difference of 50.240 ms (16.294 ms vs. 66.534 ms), as shown in Table 8. The gap widens with task complexity, peaking at 61.030 ms in S2. This pattern reflects a fundamental architectural difference: OpenClaw reasons directly toward task goals through a perceive–reason–plan–act cycle, whereas N8N processes each node sequentially with higher inter-node communication overhead. Park *et al.* (2026) identify response latency as a fundamental bottleneck in LLM-based agent deployment, making OpenClaw's advantage on this parameter a critical determinant of real-world productivity viability. Consistency tells an equally important story. OpenClaw's standard deviation in S2 was just 1.024 ms, against N8N's 116.228 ms (Table 9, Figure 6). That variance in N8N stems from a cold start problem — in S3, the first iteration recorded 252.328 ms before stabilizing around 50 ms. Ni *et al.* (2026) report that end-to-end latency optimization in agent systems can yield reductions of 1.2 to 2.4 times; this study's finding of a 4.08× average latency reduction places OpenClaw well beyond that benchmark. On resource consumption, OpenClaw used an average of 220 MB (27.7%) less memory and 14.8 percentage points (34.5%) less CPU than N8N under active conditions (Tables 10 and 11). Both gaps scale with task complexity, reaching 345 MB and 21.6% respectively in S3. Guran *et al.* (2024) stress that resource management is a critical factor in LLM deployment — and these results challenge the common assumption that LLM-based systems are inherently more resource-intensive than conventional automation platforms. A well-designed autonomous agent architecture, as demonstrated here, can be more efficient than a node-based workflow platform under equivalent workloads.

OpenClaw outperformed N8N in task accuracy across all scenarios, with an average gap of 5.6% (Table 12). The most pronounced difference appeared in S3 at 10.0%, directly confirming the hypothesis that LLM-based

adaptive reasoning provides a substantial advantage in tasks requiring dynamic decision-making. In S3, OpenClaw orchestrated multiple external APIs adaptively — adjusting execution order based on intermediate results — while N8N was constrained to a statically defined flow, making it more vulnerable to failure when conditions changed mid-execution. Zhao *et al*. (2025) identify multi-step reasoning as the primary differentiator between agentic systems and both standard LLMs and rule-based automation, precisely because agents can maintain persistent state and adjust execution strategy dynamically throughout a task's lifecycle. Yang *et al*. (2025) reported a 5.7% accuracy improvement in their AutoHMA-LLM study comparing LLM agent systems against a baseline — a figure that aligns closely with this study's 5.6% finding. That consistency across independent studies strengthens the validity of both results and suggests that the accuracy advantage of LLM-based agentic systems is a replicable phenomenon rather than an artifact of a specific implementation. It is worth noting, however, that OpenClaw's S3 accuracy of 83.3% still leaves room for improvement, particularly in tasks involving external service integration under high uncertainty.

Setup complexity is the one parameter where N8N holds a clear advantage, scoring 2.18 against OpenClaw's 3.25 on the Likert scale (Table 13). This outcome is expected: N8N was designed from the outset as a no-code platform with an intuitive visual node-graph interface, while OpenClaw requires a deeper understanding of LLM agent concepts, API configuration, and server infrastructure management. Jeong (2025) notes that the primary strength of no-code platforms lies in lowering the technical barrier to AI adoption among non-technical users, and this study confirms that advantage empirically. Broccia *et al*. (2025) add that no-code interfaces demonstrably reduce entry barriers, though at the cost of support for more complex functionality — a trade-off this study quantifies directly. The practical implication is that the choice between OpenClaw and N8N should be driven by user profile and deployment requirements. For organizations or individuals requiring high-complexity workflow automation, adaptive decision-making capability, and who possess adequate technical resources, OpenClaw is the superior option. For non-technical users who need a simple automation solution with a short implementation timeline, N8N remains the more practical choice. Dibia (2025) emphasizes the importance of human-centered design principles in agent system development, pointing to a more accessible setup experience as a priority for future OpenClaw development.

Both systems achieved a *"Good"* SUS category — OpenClaw at 78.4 and N8N at 72.6 (Table 16, Figure 13). The 5.8-point gap in OpenClaw's favor reveals an interesting dichotomy: a system that is harder to set up can still deliver a more satisfying user experience once operational. Users rated OpenClaw more favorably for its ability to understand natural language commands accurately, produce responses that reflect prior conversation history, and complete complex tasks without requiring users to define execution flows explicitly. Ependi *et al*. (2019) identify efficiency and satisfaction as two of the five core SUS dimensions, and OpenClaw's advantage in those specific dimensions likely drives its higher overall score despite a lower learnability rating. Wali *et al*. (2023) caution that SUS does not fully account for system autonomy and adaptive behavior, meaning the scores here should be read as general user satisfaction indicators rather than exhaustive measures of agent system quality. That caveat noted, the *"Good"* rating for both systems confirms that OpenClaw and N8N have each crossed the threshold of practical usability for real-world productivity settings.

The results of this study provide quantitative evidence supporting the argument that autonomous LLM-based agent systems represent a qualitative step forward relative to rule-based workflow automation platforms. OpenClaw outperformed N8N on five of six measured parameters (Table 14, Figure 11). Alva and Pandey (2026) argue that agentic AI frameworks offer software-defined flexibility through microservice-based execution and semantic task routing that node-based workflow systems cannot replicate — this study confirms that claim with measurable data. Yu *et al*. (2025) frame the transition from manually defined workflows to LLM-powered autonomous agent workflows as a significant qualitative leap; the findings here provide the empirical grounding that argument requires. Practically, this study offers concrete guidance for developers and practitioners selecting an automation architecture. Organizations that prioritize high performance, resource efficiency, and complex task handling should consider OpenClaw or equivalent agent frameworks as their primary workflow automation infrastructure. Organizations with limited technical resources that need a solution deployable quickly and with minimal configuration overhead can continue to rely on N8N as a proven option for low-to-medium complexity automation. Lei *et al*. (2025) highlight interaction steps, API calls, and token consumption as important evaluation dimensions for agent systems — and this study contributes methodologically by operationalizing those metrics within a real comparative system setting.

5 | CONCLUSIONS AND RECOMMENDATIONS

This study successfully designed, built, and evaluated an OpenClaw-based Desktop AI Assistant deployed on a Virtual Private Server, then benchmarked it empirically against N8N as an established workflow automation

baseline. Three main conclusions emerge from the full body of testing and analysis. The OpenClaw system was implemented with a functional success rate of 100% across all eight black-box test cases (Table 7). The system ran continuously for 14 days without interruption, establishing that a Docker-based VPS running Ubuntu Server 22.04 LTS is a stable and reliable deployment platform for autonomous LLM-based AI agents. All components — OpenClaw Agent, LLM Core, Telegram interface, Memory Module, Tool Connector, and monitoring stack — operated as a cohesive, self-sustaining ecosystem without constant manual intervention, as recorded in Table 5.

Performance testing shows that OpenClaw outperforms N8N on four of the six primary parameters. Response time averaged 75.5% lower (16.294 ms vs. 66.534 ms), with far greater consistency — a standard deviation of just 1.024 ms in S2 against N8N's 116.228 ms (Tables 8 and 9). Memory usage was 220 MB lower on average (573 MB vs. 793 MB), CPU utilization was 14.8 percentage points lower (28.1% vs. 42.9%), and task completion accuracy was 5.6% higher (90.0% vs. 84.4%), as shown in Tables 10, 11, and 12. These advantages grow with task complexity, indicating that LLM-based adaptive reasoning delivers greater value precisely where workflow automation demands are highest. The one parameter where N8N leads is setup complexity, scoring 2.18 against OpenClaw's 3.25 on the Likert scale (Table 13) — a reflection of N8N's visual interface advantage in initial configuration ease. The full parameter comparison is summarized in Table 14 and Figure 11.

User satisfaction evaluation using the System Usability Scale placed OpenClaw at 78.4 and N8N at 72.6, with both systems in the *"Good"* category (Table 16, Figure 13). Despite its more demanding setup, users rated the OpenClaw interaction experience more favorably — attributing this to its ability to understand natural language commands accurately and complete complex tasks without requiring users to define execution flows explicitly. Taken together, these findings confirm that autonomous LLM-based AI agent systems represent a meaningful qualitative advancement over rule-based workflow automation platforms, and warrant serious consideration as next-generation workflow automation infrastructure for organizations that prioritize high performance and complex task handling.

For future research, the scope of test scenarios should be expanded to include document processing, data analysis, and interaction with enterprise systems such as ERP and CRM platforms. A broader scenario set will produce a more complete picture of OpenClaw's capabilities and limitations in complex organizational environments. System security deserves deeper investigation in subsequent studies. Lu *et al*. (2025) identify that OpenClaw — as an agent bridging external chat applications with the local operating system — carries potential cross-environment security vulnerabilities. Penetration testing and vulnerability analysis should be conducted before the system is adopted in large-scale production environments. A more intuitive user interface for OpenClaw is strongly recommended to address its primary weakness in setup complexity. Dibia (2025) emphasizes the importance of human-centered design in agent system development, including interruptibility, capability discovery, and transparent decision-making. A web-based visual configuration interface that allows non-technical users to set up and monitor OpenClaw without deep server infrastructure knowledge would significantly improve accessibility and broaden adoption potential. Exploring local LLM models — such as Ollama running Llama or Mistral — as alternatives to cloud-based APIs like Claude or GPT-4 is also worth pursuing. This approach would reduce dependence on third-party services, lower long-term operational costs, and strengthen data privacy, particularly for organizations operating under strict data regulations in sectors such as banking, healthcare, and government. For practitioners and organizations considering adoption, a phased implementation approach is recommended — starting with low-to-medium complexity automation scenarios using N8N as a proof of concept, then migrating progressively to an OpenClaw architecture for scenarios that demand adaptive reasoning and complex task handling. A hybrid approach that combines N8N's configuration ease with OpenClaw's adaptive reasoning capability is also a promising direction worth exploring in future work.

REFERENCES

Alva, L., & Pandey, B. (2026). Agentic AI systems in the age of generative models: Architectures, cloud scalability, and real-world applications. *Artificial Intelligence Review*, *59*, 88. https://doi.org/10.1007/s10462-025-11458-6

Arzo, S. T. (2021). *Towards network automation: A multi-agent based intelligent networking system* [Doctoral dissertation, Università degli Studi di Trento].

Barra, F. L., Rodella, G., Costa, A., Scalogna, A., Carenzo, L., Monzani, A., & Corte, F. D. (2025). From prompt to platform: An agentic AI workflow for healthcare simulation scenario design. *Advances in Simulation*, *10*(1), 29. https://doi.org/10.1186/s41077-025-00357-z

Barua, S. (2024). *Exploring autonomous agents through the lens of large language models: A review* (arXiv:2404.04442). arXiv. https://doi.org/10.48550/arXiv.2404.04442

Borsci, S., Malizia, A., Schmettow, M., Van Der Velde, F., Tariverdiyeva, G., Balaji, D., & Chamberlain, A. (2022). The chatbot usability scale: The design and pilot of a usability scale for interaction with AI-based conversational agents. *Personal and Ubiquitous Computing*, *26*(1), 95–119. https://doi.org/10.1007/s00779-021-01582-9

Broccia, G., Cirillo, R., Ferrari, A., Lelii, L., & Spagnolo, G. O. (2025). *HumAInFlow: A no-code platform for modelling and simulating human-AI workflows* (Technical Report 2025/011). Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo," CNR. https://doi.org/10.32079/ISTI-TR-2025/011

Cao, J. (2024). *A study on deploying large language models as agents* [Doctoral dissertation, Massachusetts Institute of Technology]. MIT DSpace. https://hdl.handle.net/1721.1/157177

Casaclang, H., Ionescu, B., Kim, Y., & Jo, J. Y. (2026). Self-hosted workflow automation for AI-based cybersecurity operation. *Journal of The Colloquium for Information Systems Security Education*, *13*(1), 21. https://doi.org/10.53735/cisse.v13i1.241

Dibia, V. (2025). *Designing multi-agent systems: Principles, patterns and implementation for AI agents*. Victor Dibia.

Dong, H., Zhang, P., Gao, Y., Dong, X., Cheng, Y., Lu, M., & Zheng, S. (2025). *Finch: Benchmarking finance & accounting across spreadsheet-centric enterprise workflows* (arXiv:2512.13168). arXiv. https://doi.org/10.48550/arXiv.2512.13168

Ependi, U., Kurniawan, T. B., & Panjaitan, F. (2019). System usability scale vs heuristic evaluation: A review. *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, *10*(1), 65–74.

Ferrag, M. A., Tihanyi, N., & Debbah, M. (2025). *From LLM reasoning to autonomous AI agents: A comprehensive review* (arXiv:2504.19678). arXiv. https://doi.org/10.48550/arXiv.2504.19678

Guran, N., Knauf, F., Ngo, M., Petrescu, S., & Rellermeyer, J. S. (2024). *Towards a middleware for large language models* (arXiv:2411.14513). arXiv. https://doi.org/10.48550/arXiv.2411.14513

Hua, Y., Li, K., Wang, R., Liu, Y., Leng, J., Wang, G., & Yan, Y. (2025). *Towards proactive and collaborative industrial AI agents within the human-agent interaction (HAI): A review and perspective* (SSRN 6076349). SSRN. https://dx.doi.org/10.2139/ssrn.6076349

Jeong, C. (2025). Beyond text: Implementing multimodal large language model-powered multi-agent systems using a no-code platform. *Journal of Intelligence and Information Systems*, *31*(1). https://doi.org/10.13088/jiis.2025.31.1.191

Jin, W., Wang, N., Zhang, L., Tian, X., Shi, B., & Zhao, B. (2025). A review of AI-driven automation technologies: Latest taxonomies, existing challenges, and future prospects. *Computers, Materials & Continua*, *84*(3). https://doi.org/10.32604/cmc.2025.060801

Joshi, S. (2025). *Review of autonomous systems and collaborative AI agent frameworks* (SSRN 5142205). SSRN. https://ssrn.com/abstract=5142205

Koubaa, A. (2025). *Agent operating systems (Agent-OS): A blueprint architecture for real-time, secure, and scalable AI agents* (Preprint). Authorea. https://doi.org/10.20944/preprints202509.0077.v1

Lanham, M. (2025). *AI agents in action*. Simon and Schuster.

Lee, Y. S., & Moon, P. J. (2025). A comparative study of modern automation tools: Make, n8n, and Opal. *The International Journal of Advanced Smart Convergence*, *14*(4), 605–615. https://www.earticle.net/Article/A481228

Lei, Y., Xu, J., Liang, C. X., Bi, Z., Li, X., Zhang, D., & Yu, Z. (2025). *Large language model agents: A comprehensive survey on architectures, capabilities, and applications* (Preprint). Preprints. https://doi.org/10.20944/preprints202512.2119.v1

Lu, L., Gu, X., Huang, J., Du, J., Zheng, X., Liu, Y., & Pang, S. (2025). *DREAM: Dynamic red-teaming across environments for AI models* (arXiv:2512.19016). arXiv. https://doi.org/10.48550/arXiv.2512.19016

Luo, J., Zhang, W., Yuan, Y., Zhao, Y., Yang, J., Gu, Y., & Zhang, M. (2025). *Large language model agent: A survey on methodology, applications and challenges* (arXiv:2503.21460). arXiv. https://doi.org/10.48550/arXiv.2503.21460

Mahmoud, E. (2025). *Enhancing hosting infrastructure management with AI-powered automation* [Bachelor's thesis, Theseus]. https://www.theseus.fi/handle/10024/882571

Nazmunisha, N. (2026). *Multi-agent human-AI systems with low-code platforms enabling adaptive web services and real-time anomaly remediation in distributed architectures* (Preprint). Preprints. https://doi.org/10.20944/preprints202601.1877.v1

Ni, K., Hua, W., Shi, X., Guo, J., Chang, S., & Chen, T. (2026). *Chimera: Latency- and performance-aware multi-agent serving for heterogeneous LLMs* (arXiv:2603.22206). arXiv. https://doi.org/10.48550/arXiv.2603.22206

Park, G., Lee, S., & Park, Y. (2026). Minimizing response latency in LLM-based agent systems: A survey. *IEEE Access*. https://doi.org/10.1109/ACCESS.2026.3664226

Pattnayak, P., & Bohra, H. (2025). Review of tools for zero-code LLM based application development. In *World Conference on Artificial Intelligence: Advances and Applications* (pp. 294–312). Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-13803-3_24

Racharla, N. K. (2025). *Agentic AI for end-to-end investment banking automation: A no-code approach using GPT, Zapier, and Airtable* (SSRN 5263449). SSRN. https://dx.doi.org/10.2139/ssrn.5263449

Wang, X., Rosenberg, S., Michelini, J., Smith, C., Tran, H., Nyst, E., & Neubig, G. (2025). *The OpenHands software agent SDK: A composable and extensible foundation for production agents* (arXiv:2511.03690). arXiv. https://doi.org/10.48550/arXiv.2511.03690

Wali, A., Mahamad, S., & Sulaiman, S. (2023). Task automation intelligent agents: A review. *Future Internet*, *15*(6), 196. https://doi.org/10.3390/fi15060196

Xu, R., & Peng, J. (2025). *A comprehensive survey of deep research: Systems, methodologies, and applications* (arXiv:2506.12594). arXiv. https://doi.org/10.48550/arXiv.2506.12594

Yang, T., Feng, P., Guo, Q., Zhang, J., Zhang, X., Ning, J., & Mao, Z. (2025). AutoHMA-LLM: Efficient task coordination and execution in heterogeneous multi-agent systems using hybrid large language models. *IEEE Transactions on Cognitive Communications and Networking*, *11*(2), 987–998. https://doi.org/10.1109/TCCN.2025.3528892

Yu, C., Cheng, Z., Cui, H., Gao, Y., Luo, Z., Wang, Y., & Zhao, Y. (2025). A survey on agent workflow: Status and future. In *Proceedings of the 2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD)* (pp. 770–781). IEEE. https://doi.org/10.1109/ICAIBD64986.2025.11082076

Yu, J. S., & Yao, Y. (2026a). Agents and workflows. In *Intelligent language services: Theory and practice with large language models* (pp. 113–141). Springer Nature Singapore. https://doi.org/10.1007/978-981-95-4632-9_5

Yu, J. S., & Yao, Y. (2026b). Designing translation agents: Principles and practice. In *Intelligent language services: Theory and practice with large language models* (pp. 225–250). Springer Nature Singapore. https://doi.org/10.1007/978-981-95-4632-9_8

Zhang, C., He, S., Qian, J., Li, B., Li, L., Qin, S., & Zhang, Q. (2024). *Large language model-brained GUI agents: A survey* (arXiv:2411.18279). arXiv. https://doi.org/10.48550/arXiv.2411.18279

Zhao, B., Foo, L. G., Hu, P., Theobalt, C., Rahmani, H., & Liu, J. (2025). *LLM-based agentic reasoning frameworks: A survey from methods to scenarios* (arXiv:2508.17692). arXiv. https://doi.org/10.48550/arXiv.2508.17692

Zheng, Y., Chen, Y., & Hu, Y. (2025). *AudAgent: Automated auditing of privacy policy compliance in AI agents* (arXiv:2511.07441). arXiv. https://doi.org/10.48550/arXiv.2511.07441.

How to cite this article: Wali, M., Afkar, M. A., & Syafrinal, S. (2026). Design and Performance Evaluation of an OpenClaw-Based Desktop AI Assistant Deployed on Virtual Private Server: A Comparative Study with N8N Workflow Automation. *Journal Dekstop Application (JDA)*, *5*(1), 26–42. https://doi.org/10.59431/jda.v5i1.822.