IntelliDoc: An Open-Source Multi-Agent AI Builder System with Transparent Deep Research as a Use Case

Anonymous ACL submission

Abstract

We release **IntelliDoc**, an open-source multi-agent orchestration system whose **AI Builder** synthesises a complete, validated workflow graph from a single natural-language brief in one pass, enabling non-technical operators to construct, evaluate, and deploy multi-step multi-agent pipelines without writing code. A canonical use case is producing transparent, operator-tunable alternatives to commercial Deep Research products. The system exposes nine configurable agent types, each with per-agent LLM provider, model, temperature, retrieval, and tool-calling settings, and supports project-scoped multi-tenancy on shared infrastructure. ADAS (Hu et al., 2025) and AFlow (Zhang et al., 2025) search over candidate topologies and select the best against an objective; the AI Builder synthesises a single topology directly. The trade-off is explicit: no search-time optimisation, no required objective function, and a graph the operator can inspect and edit before any execution. We implement it as a four-phase pipeline (Plan, Build, Verify, Self-Critique) with a graph-algorithmic structural validator interleaved between the LLM passes. IntelliDoc adds a reference-based workflow evaluation surface with operator-supplied test datasets and an embeddable-chatbot deployment subsystem. The system is released under the MIT license.[1] Together these address seven operational limitations inherent to closed-feature Deep Research delivery.

1 Introduction

IntelliDoc is an open-source multi-agent system whose **AI Builder** generates validated, executable workflows from natural-language requests, enabling non-technical operators to automate multi-step knowledge pipelines—from document ingestion to workflow orchestration to deployed chatbot—without writing code. The architecture is domain-agnostic; we demonstrate and evaluate it here on transparent Deep Research as a representative use case. A new category of consumer LLM feature—*Deep Research*—has emerged across major providers: OpenAI Deep Research (OpenAI, 2025), Perplexity Deep Research (Perplexity AI, 2025), Gemini Deep Research (Google, 2024), Anthropic's research-mode retrieval (Anthropic, 2025), and Grok DeepSearch (xAI, 2025). All five share a common shape: a single natural-language request triggers a multi-step, multi-agent pipeline that browses sources, decomposes sub-queries, synthesises across them, and returns a citation-grounded report. Their internals are largely opaque to the operator, and their outputs are not stable across re-runs of the same query (Atil et al., 2024); IntelliDoc shares this property.

We organise the operational consequences of this opacity into a design taxonomy of seven affordances (L1–L7); each is a directly observable limitation of the five commercial systems cited above, to the best of our knowledge as of May 2026:

- **L1** cost (no per-agent control);
- **L2** inconsistency (no diagnostic surface);
- **L3** non-customisable parallel agents;
- **L4** invisible orchestration;
- **L5** model lock-in to the provider;
- **L6** opaque per-agent knowledge bases;
- **L7** no editable plan exposed to the user.

L3 and L4 are based on direct product interaction: none of the five systems exposes individual parallel-agent configuration (OpenAI, 2025; Google, 2024; Perplexity AI, 2025; Anthropic, 2025) or surfaces their internal decomposition steps to the operator (OpenAI, 2025; Anthropic, 2025). L7 most visibly distinguishes IntelliDoc from its competitors: plan-before-act is a well-established pattern (Wang et al., 2023; Yao et al., 2023), but only Gemini

[1] https://anonymous.4open.science/r/1E8F1E8F/

Deep Research surfaces an editable plan to the user; OpenAI and Anthropic's research-mode retrieval keep it internal. IntelliDoc exposes configurable mechanisms across all seven affordances: per-agent model and temperature for L1, an inspectable graph for L2 and L4, per-agent configuration for L3 and L5, per-agent retrieval settings for L6, and a per-agent Plan Mode toggle for L7.

These affordances follow from a single system-level choice: treating the workflow as an editable artefact rather than delivering Deep Research as a closed feature. The AI Builder, evaluation surface, and deployment subsystem share a single persistent graph artefact; the L1–L7 affordances are one cut of what that artefact makes possible.

IntelliDoc joins AutoGen (Wu et al., 2024), AutoGen Studio (Dibia et al., 2024), and AgentScope (Gao et al., 2024) in the line of developer-facing multi-agent frameworks. Our contributions are the **AI Builder** (§3)—a no-code natural-language interface for constructing structurally validated, operator-editable workflows—a visual canvas with nine configurable agent types (§2), a reference-based workflow evaluation surface (§5), and an embeddable-chatbot deployment subsystem (§6). The full arc from document ingestion to deployed chatbot runs on a single persistent graph artefact.

2 The IntelliDoc System

The IntelliDoc canvas is a drag-and-drop interface on which operators compose a directed workflow graph stored as JSON (Figure 1). The same graph artefact is read and written by the AI Builder, runtime executor, evaluation surface, and deployment subsystem, so workflows can be generated from natural language, refined visually, or drawn from scratch. Workflows, documents, and deployments are project-scoped, isolating each project's data from others on shared infrastructure.

2.1 Project Workspace

Each IntelliDoc project exposes a seven-page workspace: *Project Documents*, *Agent Orchestration*, *Evaluation*, *Deploy*, *Activity Tracker*, *System Performance*, and *Chatbot*. The workflow designer of Figure 1 sits inside Agent Orchestration; the Deploy page is introduced below (Figure 2) and the Project Documents page is shown in Appendix E (Figure 3). The remaining three pages—the Activity Tracker, the System Performance page, and the embeddable in-app Chatbot—are described in Appendices F, G, and H.

Project Documents. Operators upload files, folders, or zip archives and browse them as a flat list or a folder tree (Appendix E, Figure 3). A consolidated pipeline runs extraction, chunking, embedding (SentenceTransformers `all-MiniLM-L6-v2`, 384 dimensions), per-project Milvus (**?**) indexing, and optional summarisation. An *Auto Folder Classification* toggle switches the displayed folder layout between upload-derived paths and LLM-assigned paths from per-document short summaries. A separate *Full Document Mode* uploads whole documents to provider file APIs and stores the returned IDs, so nodes can attach whole files without re-chunking.

Deploy. Each workflow can be exposed as a public chat endpoint with optional username/password authentication, configurable allowed origins, per-origin rate limits, and a copy-paste public URL (Figure 2). Both non-streaming and Server-Sent-Events endpoints are exposed, and conversation context is persisted in a per-deployment session so a 429 rate-limit response does not lose prior turns. Operational details are described in §6.

2.2 Agent Palette

Table 1 summarises the nine agent types. The non-obvious ones are the **Classifier** (1-of-N routing, one outgoing edge per category), the **Splitter** (parallel distribution across capability-matched specialists; unselected specialists are pruned at runtime; specialist failures narrow output coverage but do not halt the workflow), and the **MCP Server** (a node that exposes an external Model Context Protocol (Anthropic, 2024) server as a tool surface to upstream agents; node failures are reported to the invoking agent without halting execution; Google Drive and SharePoint are included by default). Failed LLM calls are retried up to three times before the error is passed to the downstream agent.

2.3 Per-Agent Capabilities

Per-agent provider, model, temperature, and role settings (Table 2) give operators fine-grained cost levers for L1, per-agent diversity for L3, and provider flexibility for L5.

Retrieval is independently togglable across three channels (Lewis et al., 2020). *Document RAG* retrieves project-scoped chunks at inference time, with folder and file-ID filters and seven search strategies including hierarchical retrieval (Sarthi

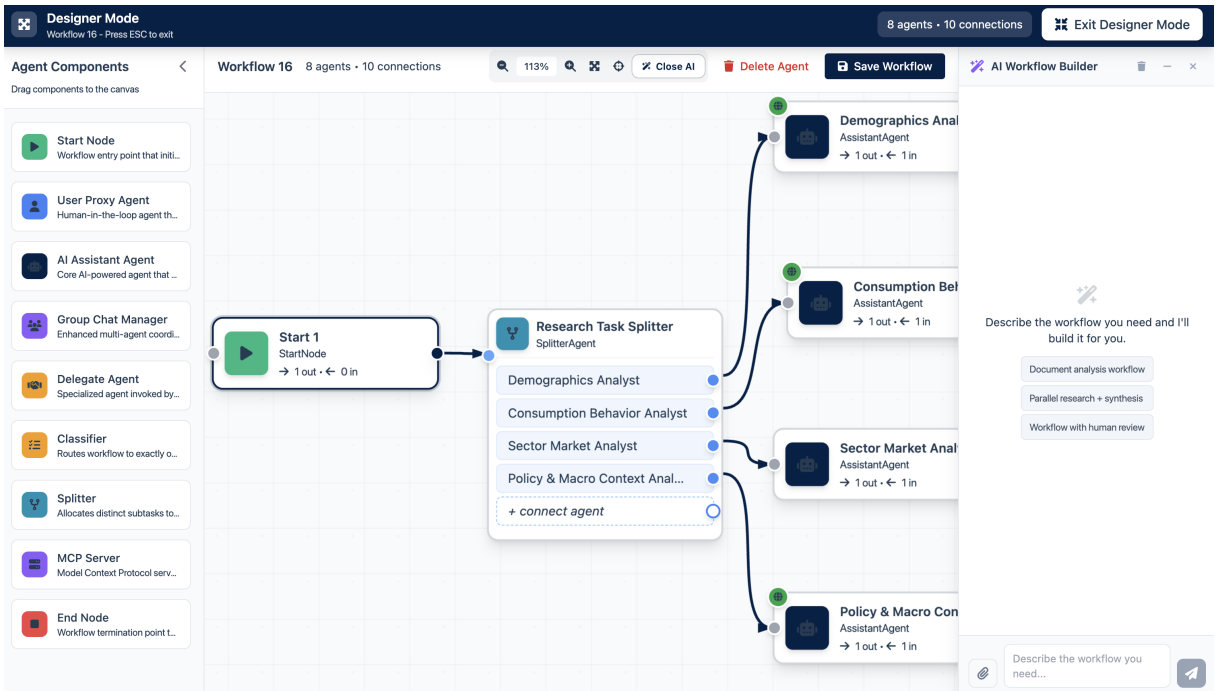

Figure 1: The IntelliDoc canvas: the agent palette on the left, an AI-Builder-authored workflow in the centre, and the AI Builder chat panel on the right.

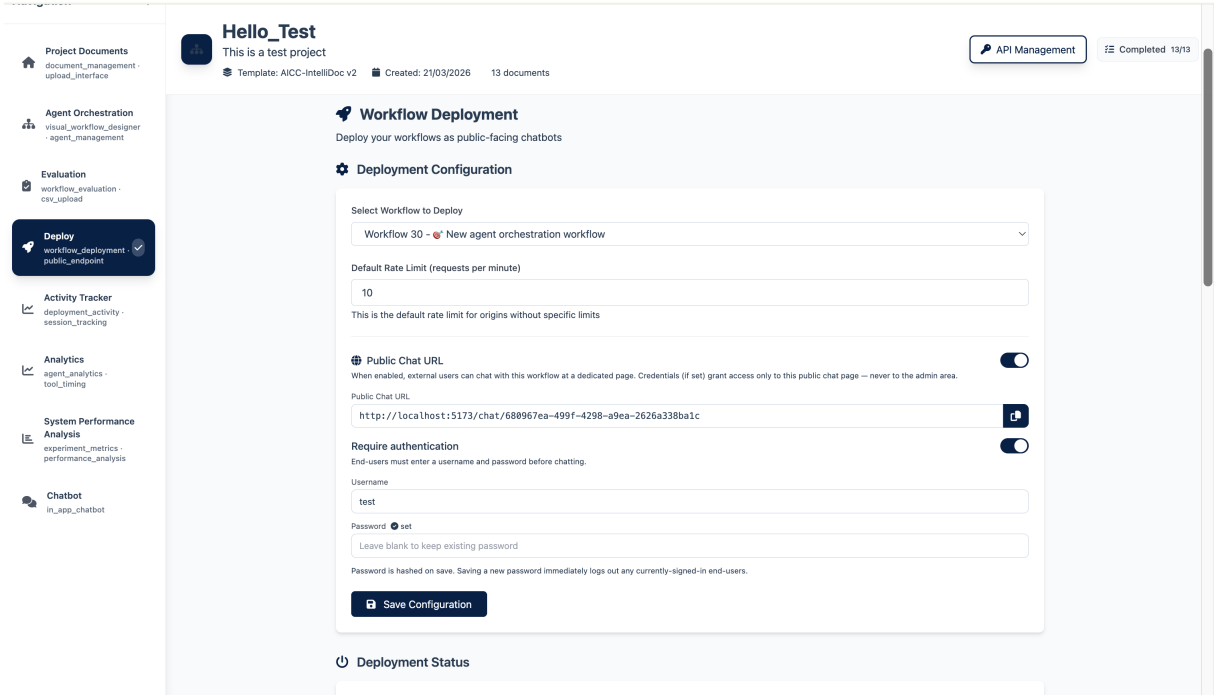

Figure 2: The Deploy page: workflow-deployment configuration form for the public chat URL, authentication, and rate limits.

et al., 2024). *Document Tool Calling* exposes typed queries against the project store as a tool call, with an optional Plan Mode that previews which documents will be read (L7). *Web Search* operates in three modes (general, domain-list, URL-list), with excerpts cached across runs to amortise per-query cost (L1).

Visual workflow builders share a drag-and-drop authoring surface that lowers the multi-agent prototyping barrier (Dibia et al., 2024; Gao et al., 2024), but treat natural language differently. Flowise (FlowiseAI, 2024) and Langflow (DataStax, 2024) ex-

Agent type	Primary role
Start	Workflow entry; user prompt is captured here.
AI Assistant	Primary LLM agent; per-agent retrieval, web search, and tool calls.
User Proxy	User input node; first-class retrieval and web search; pauses for human review when configured.
Group Chat Manager	Multi-speaker coordination inside a branch.
Delegate	Nested delegation to a manager agent.
Classifier	1-of-N routing on the incoming message.
Splitter	Parallel allocation across capability-matched specialists.
MCP Server	Tool surface for an external Model Context Protocol server.
End	Workflow exit; configurable output format.

Table 1: The nine agent types in the IntelliDoc palette.

Setting	Per-agent options
LLM provider	OpenAI, Anthropic, Google
Model	Any provider-supported model
Temperature	0–2 (set $T = 0$ for determinism)
Agent role	Free-form text describing the agent's behaviour
Document RAG	7 search strategies; folder / file-ID filters
Doc Tool Calling	Optional Plan Mode preview (L7)
Web Search	General / domain-list / URL-list modes
Reflection edges	Opt-in back-edges with an iteration cap

Table 2: Per-agent configuration surface exposed by every *AI Assistant*, *Delegate*, *User Proxy*, and *Group Chat Manager* node.

pose chat panels that author or modify individual components within an otherwise hand-built graph. Dify (Dify Team, 2024) uses NL prompts to shape node behaviour inside app-style RAG and chat templates rather than to synthesise the topology. LangGraph Studio (LangChain AI, 2024) is an inspection and debug surface over a code-defined graph rather than an NL-authored one. AutoGen Studio (Dibia et al., 2024) is a no-code tool for composing agent teams from a typed palette: each agent requires individual configuration (role, model, tool assignments) and edges must be wired manually; there is no NL-driven topology synthesis. The differentiator is not the palette—IntelliDoc has one too—but the AI Builder's ability (§3) to accept a single natural-language brief and produce a fully wired, structurally validated workflow in one pass. None of the five systems closes this gap.

3 The AI Builder

The AI Builder is the system's core contribution: it synthesises a deterministically validated, operator-editable workflow graph from a single natural-language request in one pass. ADAS (Hu et al., 2025) and AFlow (Zhang et al., 2025) search over candidate topologies, selecting the best-performing variant against a defined objective at search-time cost; the AI Builder requires no such budget or objective, and produces a graph the operator can inspect and edit before execution. By contrast, MetaGPT (Hong et al., 2024) hard-codes role-based standard operating procedures (SOPs) in source rather than synthesising them from natural language. We implement the AI Builder as a four-phase pipeline (Plan, Build, Verify, Self-Critique) that gates the transition from LLM authoring to LLM refinement with a deterministic structural validator. Each phase operates on the same in-memory graph, so later phases observe the cumulative effect of earlier ones.

Phase 1 (Plan). *Text-only LLM call; no tools.* The planner emits a numbered list of agent additions, updates, and removals plus the final flow `Start` $\rightarrow$ . . . $\rightarrow$ `End`.

Phase 2 (Build). *Tool-calling LLM; 17-tool authoring API (Appendix B); up to three retries per rejected call.* Tools fall into node creators (one per agent type except MCP Server, which operators place manually), edge wiring, property updates, and deletions. Arguments are validated against the API schema; rejected calls are returned to the Build LLM as structured errors, and unresolved requests carry forward to Phase 3.

Phase 3 (Verify). *Deterministic invariant checker first; LLM verifier on residuals plus light semantic refinement, $\leq$ 6 iterations.* The checker enforces five graph-decidable invariants, each linear in the number of nodes and edges ($O(|V| + |E|)$): (i) exactly one Start and one End; (ii) acyclicity on sequential edges via Kahn's algorithm (Kahn, 1962); (iii) every classifier category wired to $\geq$ 1 outgoing edge; (iv) every Splitter wired to $\geq$ 2 downstream specialists; (v) every non-terminal agent reachable from Start and reaching End. Residuals are handed to the verifier LLM for surgical repair.

Phase 4 (Self-Critique). *LLM critic on the same 17-tool surface, $\leq$ 4 iterations.* The critic targets semantic issues a deterministic checker cannot see: vague role descriptions, redundant agents, provider–

model mismatches, downstream specialists without distinct roles. The loop exits on a zero-tool-call response. Critic deletions are not undone by the structural repair step, so Phase 4 cannot reverse Phase 3's structural decisions; operators can override on the canvas if a Phase 3 repair was inappropriate for the request.

Why interleave deterministic and LLM phases. Gating the LLM phases with a deterministic structural check is what separates this design from iterative-search alternatives: well-formedness is decided by graph-theoretic checks and cheap to repair surgically, while only semantic adequacy is left to an LLM critic. Runtime correctness remains an empirical question, addressed in §4.

4 Empirical Study

Scope. The study measures two complementary aspects of the AI Builder: *synthesis-side* behaviour (structural validity of synthesised graphs, phase iteration counts, and topology distribution) and *downstream output quality* (RACE scores of the resulting reports against DeepResearch Bench references).

Prompt set and setup. We evaluate on 30 prompts from DeepResearch Bench (Du et al., 2025) spanning six topic areas: Finance & Business, Science & Technology, Software Development, Education & Jobs, Health, and Literature (full benchmark description in Appendix A). Each prompt is submitted to the AI Builder with the wrapper directive *"Generate a deep research workflow to investigate the below:"* and processed under `gpt-5.3-chat-latest` at temperature 0; prompts 66–80 additionally carry an explicit length directive (*"approximately 5,000 words"*) to match reference article length. The AI Builder is invoked once per prompt, producing 30 distinct workflows (Figure 1). **No operator manually edited any synthesised workflow** before execution—every measurement below reflects fully automatic synthesis and establishes a fully automatic baseline. Prompts 51–65 received a full per-prompt audit: each synthesised graph was individually verified against the five structural invariants, per-phase synthesis metrics were collected, each runtime execution trace was manually inspected, and each judge response was validated before inclusion. Prompts 66–80 were generated via the same pipeline, validated by successful end-to-end execution in every case (a weaker form of validation than post-hoc graph inspection), and evaluated by the same RACE judge pipeline. Per-prompt measurements are tabulated in Appendix D (Table 8); graph JSONs, outputs, and aggregation scripts ship with the open-source release.

Synthesis structural validity and AI Builder behaviour. All $\mathbf{30/30}$ synthesised workflows satisfy the five structural invariants of §3: 15/15 confirmed by post-hoc evaluation of saved graph JSONs (prompts 51–65; Table 3), and 15/15 validated by end-to-end execution (prompts 66–80), a weaker form of validation than post-hoc graph inspection. The deployed validator is the five-invariant set; the system also runs a stricter internal heuristic in parallel that surfaces additional warnings to the Verify LLM but does not block synthesis. The heuristic disagreed with the post-hoc invariant evaluation on $1/15$ graphs (6.7%), a false-positive that the post-hoc evaluator confirms is not a violation of invariants (i)–(v) of §3; that workflow executed correctly at runtime. The verifier-LLM is semantic-active in 35.7% of first-pass-valid runs (typically a property-update tool call to expand agent system messages), confirming that Phase 3 in practice combines structural verification with light semantic refinement. The invariants are sound on all 30 synthesised graphs; completeness—whether (i)–(v) are sufficient, not merely necessary, for runtime correctness—is not formally claimed; observed runtime failures during development have consistently mapped to invariant violations rather than to graphs satisfying all five.

Topology and per-agent configuration. Table 4 reports topology ($N = 30$) and feature-flag distribution ($N = 15$; prompts 51–65). The AI Builder produced predominantly Splitter-based designs with 4.6 *AssistantAgent*s on average. In $26/26$ runs with a Splitter, the cheaper `gpt-4o-mini` model was selected for routing while analysts used `gpt-5.3-chat-latest`: the AI Builder selects different models per agent role rather than a single planner default, consistent with the cost-control mechanism (L1) described in §2.3; this reflects an explicit instruction in the planner prompt rather than emergent model selection.

Runtime execution. Table 5 reports per-execution behaviour across all 30 runs. Parallel execution was used in $27/30$ runs (90%) and web search in $27/30$ (90%), consistent across both prompt batches. For the audited subset (prompts 51–65), the Splitter pruned 26.2% of candidate specialists at runtime (11 of 42 offered) as not

Metric	Value	95% CI
Structural validity		
First-pass-valid, system heuristic	14/15 (93.3%)	[70.2, 98.8%]
First-pass-valid, 5-invariant (post-hoc)	15/15 (100%)	[79.6, 100%]
Final valid, 5-invariant (post-hoc)	15/15 (100%)	[79.6, 100%]
Phase 3 verifier-LLM		
Mean iterations / run	1.93	—
Iteration cap reached (≥ 6)	1/15	—
Mean tool calls / run	1.07	—
Semantic-active on valid graphs	5/14 (35.7%)	[16.3, 61.2%]
Phase 4 Self-Critique		
Runs with ≥ 1 tool call	13/15 (86.7%)	[62.1, 96.3%]
Mean iterations / run	3.00	—
Mean tool calls / run	2.33	—

Table 3: AI Builder synthesis behaviour on the individually audited $N = 15$ prompts (51–65). Wilson 95% intervals where applicable.

Property	Count / value
Topology (per workflow, $N = 30$)	
Splitter-based	26/30 (87%)
Linear	4/30 (13%)
Mean *AssistantAgent*s / workflow	4.6
Feature flags (≥ 1 analyst enabled, $N = 15$)†	
Web Search	14/15 (93%)
Document Tool Calling	14/15 (93%)
Plan Mode (L7)	5/15 (33%)
Document RAG (always-on)	0/15
Model-tier diversity / L1 ($N = 30$)	
Splitter using `gpt-4o-mini`	26/26
Analyst using `gpt-5.3-chat-latest`	138/138 agents

Table 4: Topology and model-tier selection across $N = 30$ workflows. †Feature-flag data collected for prompts 51–65 only.

Metric	Value
End-to-end duration (s)	mean 97.8, range 36.2–199.3
Runs with parallel execution	27/30 (90%)
Mean Splitter fan-out	3.4 candidates‡
Specialists pruned at runtime	11/42 (26.2%)†
Runs with ≥ 1 web search	27/30 (90%)
Mean web searches / run	3.2

Table 5: Runtime execution behaviour across $N = 30$ end-to-end runs. †Pruning data from prompts 51–65 only. ‡Synthesis-time fan-out for prompts 66–80; runtime alloc for prompts 51–65.

capability-matched for the incoming subtask, confirming the runtime allocation behaviour described in §2.2.

Execution-time comparison with commercial Deep Research. Based on vendor-reported typical-case ranges, OpenAI Deep Research (OpenAI, 2025) takes 5–30 minutes per query, Gemini Deep Research (Google, 2024) 8–10 minutes, Anthropic Deep Research (Anthropic, 2025) 5–15 minutes, and Perplexity Deep Research (Perplexity AI, 2025) 2–4 minutes. IntelliDoc's end-to-end execution ranged from 36 to 199 seconds (mean 97.8 s): prompts 51–65 averaged 75.7 s; prompts 66–80 (with the length directive) averaged 119.9 s. Even the slowest run falls within or below Perplexity's 2–4 minute range and well under the OpenAI, Gemini, and Anthropic ranges. Synthesis itself adds a one-time ∼30–60 s cost that is amortised across re-executions of the same workflow on new inputs.

End-to-end report quality (RACE). We score each of the 30 IntelliDoc reports against the corresponding DeepResearch Bench reference on the RACE rubric (Du et al., 2025) (four weighted dimensions, 0–10; primary judge `gpt-5.3-chat-latest`). The references are Gemini-2.5-Pro Deep Research outputs from April 2025 (Du et al., 2025), making the IntelliDoc/Reference ratio a direct quality comparison against a commercial Deep Research product. Table 6 shows an overall ratio of **88.9**%, with Readability near-parity (96.7%) and Comprehensiveness as the primary gap (82.5%). The explicit length directive carried by prompts 66–80 was associated with a narrower Comprehensiveness gap (85.5% vs 79.6% for prompts 51–65); the groups also differ in domain, so length and domain shift are jointly confounded. Together with

the execution-time results, IntelliDoc offers a faster, more transparent alternative at 88.9% of reference quality. Under DRB's normalised scoring formula (Du et al., 2025), IntelliDoc scores 47.1 under our judge; for context, published leaderboard scores are OpenAI Deep Research 46.98 and Gemini-2.5-Pro Deep Research 48.88, though those scores use DRB's leaderboard judge (Gemini-2.5-Pro Preview) and are not directly comparable.

Cross-family judge replication. To bound same-family judge bias, we re-ran the same 30 IntelliDoc reports and DeepResearch Bench references through an independent judge from a different model family (`claude-opus-4-7`, Anthropic). Per-prompt overall IntelliDoc scores correlate at Pearson $r = 0.84$ (Kendall $\tau = 0.63$) between the two judges, and the per-prompt IntelliDoc/Reference ratio correlates at $r = 0.70$ ($\tau = 0.43$). `claude-opus-4-7` is the stricter judge (Table 7), with an aggregate ratio of 76.3% versus 88.9% under the primary judge. The lower Reference correlation ($r = 0.45$) reflects calibration differences on reference scoring, not instability in relative ordering. The Comprehensiveness dimension takes the largest absolute hit under the stricter judge, consistent with the length-driven gap above; the directional finding holds across both families, though absolute magnitude is judge-dependent.

Limitations specific to this study. (1) Evaluation covers 30 prompts from DeepResearch Bench across six topic areas; broader coverage is left to future work. (2) Single planner LLM (`gpt-5.3-chat-latest`); behaviour under a different planner is unmeasured. (3) The primary RACE judge was the same model family as the planner; Table 7 reports the cross-family replication. (4) Per-prompt repetitions are 1; cross-repetition variance is not reported. Appendix C traces one end-to-end run.

5 Evaluation

IntelliDoc supports supervised evaluation over operator-supplied test datasets. Each instance pairs a natural-language input with a reference answer; the workflow's terminal output is scored using ROUGE-1/2/L (Lin, 2004), sentence-level BLEU (Papineni et al., 2002), BERTScore (Zhang et al., 2020), and embedding cosine similarity (Reimers and Gurevych, 2019), with the mean serving as a summary scalar; these n-gram and embedding metrics are known to be weak signals for long-form outputs. The evaluation is metric-pluggable: the same dataset hooks accept LLM-as-judge scorers of the kind used for the RACE evaluation in §4, as well as faithfulness and citation-precision scorers.

Test datasets are loaded as CSV or JSON and are scoped to a project. Operators can compare workflow variants on the same labelled split, which makes regression testing across edits possible without re-running the production workflow.

6 Deploy

The Deploy subsystem exposes each authored workflow as a public, embeddable chatbot at its own URL (Figure 2), with a JSON API for programmatic access. Operators embed it as an `<iframe>` widget or call the API directly. Each deployment configures optional password authentication, per-origin CORS, rate limiting, and structured citation rendering.

Sessions and aggregate timing/volume statistics surface in a per-project dashboard. The same persistent graph artefact serves authoring, evaluation, and deployment, so operators do not rebuild them per hosting target.

7 Discussion

Open-Source Release. IntelliDoc is an open-source multi-agent orchestration system: operators construct validated workflows from natural language via the AI Builder (§3), evaluate output quality against labelled datasets (§5), and deploy the same workflow as a persistent, embeddable chatbot (§6). The system is MIT-licensed; a `docker-compose` bundle gives one-step local installation.

Operator transparency in deployment. The per-agent property inspection (§2.1), together with the Activity Tracker and System Performance page (Appendices F and G), log per-session traces of deployed workflow behaviour so operators can audit it over time. The same graph artefact authored by the AI Builder and refined on the canvas remains visible at execution time, so post-deployment behaviour is auditable rather than hidden behind a chat interface.

Conclusion. IntelliDoc is an open-source multi-agent AI Builder system synthesising operator-editable workflows from natural language, spanning document ingestion to deployed chatbot without writing code. It commits to one-pass, structurally validated synthesis on a graph the operator

Dimension	IntelliDoc	Reference	Gap	Ratio
Comprehensiveness	7.61	9.22	-1.61	82.5%
Insight/Depth	8.12	8.99	-0.87	90.3%
Instruction-following	8.58	9.44	-0.86	90.9%
Readability	8.65	8.94	-0.29	96.7%
Overall (weighted)	**8.13**	**9.15**	$\mathbf{-1.02}$	**88.9%**

Table 6: Mean RACE scores (0–10) for IntelliDoc and DeepResearch Bench reference reports across $N = 30$ prompts. **Overall** is the dimension-weighted mean across prompts. **Ratio** = IntelliDoc / Reference. Bootstrap 95% CI for Overall ratio: $[87.3\%, 90.4\%]$.

	Primary judge: gpt-5.3-chat-latest			Cross-family judge: claude-opus-4-7		
Dimension	**IntelliDoc**	**Reference**	**Ratio**	**IntelliDoc**	**Reference**	**Ratio**
Comprehensiveness	7.61	9.22	82.5%	5.92	8.66	68.4%
Insight/Depth	8.12	8.99	90.3%	6.28	8.20	76.6%
Instruction-following	8.58	9.44	90.9%	7.32	8.93	82.0%
Readability	8.65	8.94	96.7%	7.17	8.48	84.6%
Overall (weighted)	**8.13**	**9.15**	**88.9%**	**6.51**	**8.53**	**76.3%**

Table 7: Cross-family judge replication of RACE on the same $N = 30$ IntelliDoc reports and DeepResearch Bench references. **Cross-judge agreement** on per-prompt overall scores: Pearson $r = 0.84$, Kendall $\tau = 0.63$ on IntelliDoc; $r = 0.45$, $\tau = 0.37$ on Reference; $r = 0.70$, $\tau = 0.43$ on the per-prompt ratio.

can inspect and edit before execution, with evaluation and deployment over the same artefact. Transparent Deep Research is a canonical use case; the same pipeline applies to any multi-step knowledge workflow an operator can describe in natural language. The AI Builder is faster and more transparent than commercial Deep Research, with a measurable comprehensiveness gap on long-form outputs (Tables 6 and 7); operators gain inspectable, configurable, redeployable pipelines.

Limitations

§4 reports synthesis-side, runtime-side, and end-to-end-quality measurements on $N = 30$ prompts under a single planner LLM, with study-specific limitations enumerated in that section. An operator-refined-workflow comparison is deferred to a follow-up study. Output quality is bounded by the planner LLM's capability; the model-agnostic architecture lets operators swap in a stronger planner but does not lift that limit. The four-phase pipeline design is motivated by observed phase behaviour: Phase 3 LLM verifier is semantic-active in 35.7% of valid graphs; Phase 4 issues tool calls in 86.7% of runs. A controlled per-phase ablation is deferred to future work. The no-code authoring framing is a design goal; empirical validation via user study is deferred.

Ethical Considerations

IntelliDoc lowers the barrier for non-technical operators to deploy multi-agent knowledge workflows to end users; AI-generated outputs should be reviewed before use in consequential decisions. Three mechanisms partially mitigate deployment risk: the evaluation surface (§5) supports regression testing of workflow variants on labelled splits before redeployment; the Activity Tracker (Appendix F) preserves per-session transcripts and tool-call traces as an audit log over deployed traffic; and per-project encrypted API-key storage prevents cross-tenant credential leakage on shared deployments. These are mitigations, not substitutes for human review of consequential outputs.

References

Anthropic. 2024. Model context protocol. https://modelcontextprotocol.io/. Accessed: 2026-05-13.

Anthropic. 2025. Claude can now search the web. https://www.anthropic.com/news/web-search. Web-search and "Research" multi-step retrieval modes on Claude. Accessed 2026-05-12.

Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin.

2024. Non-determinism of "deterministic" LLM settings. https://arxiv.org/abs/2408.04667. ArXiv:2408.04667; latest revision 2025-04-02.

DataStax. 2024. Langflow documentation: A visual framework for building multi-agent and RAG applications. https://docs.langflow.org/. Accessed: 2026-05-13.

Victor C. Dibia, Jingya Chen, Gagan Bansal, Suff Syed, Adam Fourney, E. Zhu, Chi Wang, and Saleema Amershi. 2024. AutoGen Studio: A no-code developer tool for building and debugging multi-agent systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*.

Dify Team. 2024. Dify documentation: Open-source LLM application development platform. https://docs.dify.ai/. Accessed: 2026-05-13.

Mingxuan Du, Benfeng Yang, Zhao Liao, Wenhao Han, Wei Yuan, Yangzhou Sun, Tianyu Liu, Lu Hou, and Tat-Seng Chua. 2025. Deep-research bench: A comprehensive benchmark for deep research agents. https://arxiv.org/abs/2506.11763. ArXiv:2506.11763. 100 PhD-level research tasks across 22 domains, with the RACE and FACT evaluation frameworks. Repository: https://github.com/Ayanami0730/deep_research_bench. Accessed 2026-05-14.

FlowiseAI. 2024. Flowise documentation: Drag & drop UI to build LLM flows. https://docs.flowiseai.com/. Accessed: 2026-05-13.

Dawei Gao, Zitao Li, Weirui Kuang, Xuchen Pan, Daoyuan Chen, Zhijian Ma, Bingchen Qian, Liuyi Yao, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. AgentScope: A flexible yet robust multi-agent platform. arXiv preprint arXiv:2402.14034.

Google. 2024. Try deep research and our new experimental model in Gemini. https://blog.google/products/gemini/google-gemini-deep-research/. Released 2024-12-11 for Gemini Advanced subscribers; built on Gemini 2.0 with a multi-step planning and browsing pipeline. Accessed 2026-05-12.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*. Oral presentation, top 1.2%.

Shengran Hu, Cong Lu, and Jeff Clune. 2025. Automated design of agentic systems. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*.

A. B. Kahn. 1962. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

LangChain AI. 2024. LangGraph documentation: Building stateful, multi-actor applications with LLMs. https://langchain-ai.github.io/langgraph/. Accessed: 2026-05-13.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

OpenAI. 2025. Introducing deep research. https://openai.com/index/introducing-deep-research/. Released 2025-02-02. Agentic web-research mode on ChatGPT powered by an o-series model; runs multi-step research and returns a single synthesised report. Accessed 2026-05-12.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Perplexity AI. 2025. Introducing perplexity deep research. https://www.perplexity.ai/hub/blog/introducing-perplexity-deep-research. Released 2025-02-14. Iterative search-read-reason agent on the Perplexity platform; returns a report grounded in cited sources. Accessed 2026-05-12.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. arXiv preprint arXiv:1908.10084.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations (ICLR)*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Proceedings of the First Conference on Language Modeling (COLM)*.

xAI. 2025. Grok DeepSearch: Multi-step reasoning search. https://x.ai/news/grok-3. DeepSearch and "Big Brain" multi-step reasoning modes shipped with Grok 3 (February 2025). Accessed 2026-05-12.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2025. AFlow: Automating agentic workflow generation. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR)*.

A DeepResearch Bench: Dataset Details

DeepResearch Bench (Du et al., 2025) is a benchmark of 100 PhD-level research tasks crafted by domain experts across 22 fields, with topical distribution derived from analysis of 96,147 anonymised real user queries. The benchmark provides per-prompt reference articles and expert-written evaluation criteria across four weighted dimensions (Comprehensiveness, Insight/Depth, Instruction-following, Readability).

We evaluate on 30 prompts from the English subset (IDs 51–80): Finance & Business (51–57), Science & Technology (58–65), Software Development (66–70), Education & Jobs (71–74), Health (75–78), Literature (79–80). The two-tier audit design (prompts 51–65 individually audited; 66–80 validated by end-to-end execution) is described in §4.

B The 17-Tool Authoring API

The 17 tools, grouped by category:

- **Node creators (8)** — one per agent type, except MCP Server which operators place manually on the canvas: `add_start_node`, `add_assistant_agent`, `add_user_proxy_agent`, `add_group_chat_manager`, `add_delegate_agent`, `add_classifier_agent`, `add_splitter_agent`, `add_end_node`.
- **Edge wiring (3)** — `connect_nodes` (classifier edges require a category handle, delegate edges are auto-typed, reflection edges carry an iteration cap), `rewire_edge`, `change_edge_type`.
- **Property updates (3)** — `update_node_property`, `add_category`, `remove_category`.
- **Deletions (3)** — `delete_node`, `delete_edge`, `clear_all_agents`.

C End-to-End Walkthrough

This walkthrough traces the natural-language request *"Build me a workflow that researches a competitor, summarises their pricing, and drafts a customer-facing comparison"* through all four phases (§3).

Phase 1 (Plan). The planner emits a numbered plan: (i) add a Splitter that distributes work between a Source Researcher (web search on competitor pricing) and a Pricing Analyst (Document RAG over internal pricing files); (ii) add a Drafter that synthesises both outputs into a customer-facing comparison; (iii) final flow `Start` → `Splitter` → `{Source Researcher, Pricing Analyst}` → `Drafter` → `End`.

Phase 2 (Build). The Builder issues, in sequence, `add_splitter_agent`; two `add_assistant_agent` calls (one with web search enabled, one with document RAG enabled); a third `add_assistant_agent` for the Drafter; and four `connect_nodes` calls wiring the directed acyclic graph (DAG).

Phase 3 (Verify). The deterministic checker confirms the graph is acyclic on sequential edges, the Splitter has two downstream specialists (satisfying the fan-out invariant), and every non-terminal agent has a path to End. No verifier-LLM intervention is required for this request.

Phase 4 (Self-Critique). The critic detects that the two specialists have only bullet-style names as role descriptions, too vague for the Splitter to differentiate them reliably. It issues `update_node_property` on each to expand the descriptions into distinct responsibility statements; the next iteration emits zero tool calls and the loop exits.

D Per-Prompt Audit Table

Table 8 reports every per-prompt measurement underlying §4: synthesised topology, final-graph node

and edge counts, runtime duration, Splitter allocation and pruning at execution, web search count, and the RACE overall scores for IntelliDoc and the DeepResearch Bench reference. The aggregation script and underlying data files are included in the open-source release.

Model snapshots. All AI Builder syntheses, workflow executions, and the primary RACE-judge calls reported in §4 used the OpenAI API alias `gpt-5.3-chat-latest` accessed during the May 2026 ARR submission window (7–19 May 2026). The cross-family RACE judge in Table 7 used the Anthropic API alias `claude-opus-4-7` accessed in the same window. Both are production aliases; the open-source release ships per-call request and response payloads for independent replication.

E Project Documents

The Project Documents page (Figure 3) supports upload of files, folders, or zip archives, browsable as a flat list or folder tree. The processing pipeline runs extraction, chunking, embedding, per-project Milvus indexing, and optional summarisation; an *Auto Folder Classification* toggle switches the displayed folder layout between upload-derived and LLM-assigned paths.

F Activity Tracker

The Activity Tracker is the operator's view over deployed conversations (Figure 4). Each row is a session with message count, started timestamp, and last-activity timestamp; rows expand to show the full turn-by-turn transcript and tool-call traces. Filtering by session ID supports drill-down when an external user reports a specific issue.

G System Performance

The System Performance page reports daily success/failure counts and a user-query word cloud across all deployment sessions (Figure 5). Per-run metrics include per-agent message counts, latencies, and token usage. The underlying experiment-metric rows are the source of the runtime numbers reported in §4. **Privacy controls for the word cloud.** The cloud aggregates query terms across all deployment sessions of a project after a configurable retention window; operators can disable the word cloud per deployment via the Deploy configuration, and the underlying session transcripts are stored only for the retention period configured at deployment time. The Activity Tracker (Appendix F) provides the per-session audit trail; the word cloud is a coarse-grained aggregate of the same data after expiry of finer-grained transcripts.

H In-app Chatbot

The in-app Chatbot is an embeddable conversational interface inside the project (Figure 6). A left-hand sidebar lists prior conversations per (project, deployment) pair, with create, rename, and delete actions; conversation state is persisted in `localStorage` under a stable per-project key. A fullscreen mode and `postMessage`-based iframe integration allow the same chat to be embedded into external host pages with the workflow running unchanged behind it.

ID	topic	topo	n	e	dur(s)	alloc	prune	web	RACE	ref	ratio
51	Fin&Bus	Split	8	10	69.8	3	1	4	7.31	9.21	79%
52	Fin&Bus	Linear	5	4	36.2	0	0	0	7.38	9.20	80%
53	Fin&Bus	Linear	7	8	89.9	0	0	6	7.92	8.95	89%
54	Fin&Bus	Split	7	7	88.8	2	0	2	8.47	9.43	90%
55	Fin&Bus	Linear	8	13	38.6	0	0	1	7.59	9.05	84%
56	Fin&Bus	Split	8	10	44.6	2	2	0	8.08	8.79	92%
57	Fin&Bus	Split	7	8	98.4	3	0	5	8.10	8.93	91%
58	Sci&Tech	Split	7	8	59.7	2	1	1	7.64	9.33	82%
59	Sci&Tech	Split	7	8	93.3	3	0	5	8.54	9.38	91%
60	Sci&Tech	Split	8	10	69.5	1	3	1	8.06	9.04	89%
61	Sci&Tech	Split	7	8	100.8	3	0	6	7.83	8.09	97%
62	Sci&Tech	Split	7	8	97.8	2	1	3	8.00	9.25	87%
63	Sci&Tech	Split	8	10	94.0	2	2	4	8.07	9.01	90%
64	Sci&Tech	Split	8	10	68.7	3	1	5	8.36	9.23	91%
65	Sci&Tech	Split	9	12	84.7	5	0	1	8.83	8.98	98%
66	Soft.Dev	Split	8	9	148.7	$2^{\dagger}$	—	4	8.16	9.29	88%
67	Soft.Dev	Split	7	8	199.3	$3^{\dagger}$	—	3	8.18	9.19	89%
68	Soft.Dev	Split	7	8	105.2	$3^{\dagger}$	—	2	7.81	9.12	86%
69	Soft.Dev	Split	7	7	139.4	$2^{\dagger}$	—	4	8.45	9.37	90%
70	Soft.Dev	Split	8	10	99.3	$4^{\dagger}$	—	4	8.75	9.34	94%
71	Edu/Jobs	Split	8	10	95.8	$4^{\dagger}$	—	3	8.71	9.21	95%
72	Edu/Jobs	Linear	7	9	193.1	—	—	2	7.73	9.13	85%
73	Edu/Jobs	Split	7	8	76.3	$3^{\dagger}$	—	0	8.56	9.40	91%
74	Edu/Jobs	Split	7	8	91.4	$3^{\dagger}$	—	3	8.36	9.09	92%
75	Health	Split	8	10	101.9	$4^{\dagger}$	—	5	8.00	9.35	86%
76	Health	Split	9	12	101.0	$5^{\dagger}$	—	6	8.36	9.36	89%
77	Health	Split	8	10	103.7	$4^{\dagger}$	—	3	8.56	9.33	92%
78	Health	Split	7	8	96.9	$3^{\dagger}$	—	3	8.54	9.22	93%
79	Lit.	Split	8	10	140.3	$3^{\dagger}$	—	8	7.48	9.24	81%
80	Lit.	Split	7	8	106.4	$3^{\dagger}$	—	3	8.21	9.06	91%
total					—	**77**	**$11^{\dagger}$**	**97**	—	—	—
mean					**97.8**	—	—	—	**8.13**	**9.15**	**89%**

Table 8: Per-prompt audit table for all $N = 30$ prompts. **topo**: Split = Splitter-based, Linear = no Splitter. **n/e**: final-graph node and edge counts. **dur(s)**: end-to-end execution duration. **alloc**: runtime allocation (51–65) or synthesis fan-out† (66–80). **prune**: specialists pruned at runtime (prompts 51–65 only). **web**: web search calls. **RACE/ref**: IntelliDoc and DeepResearch Bench reference overall RACE scores (0–10) under the primary judge. **ratio**: IntelliDoc / reference. The **total** row sums count columns; **mean** averages score and duration columns.

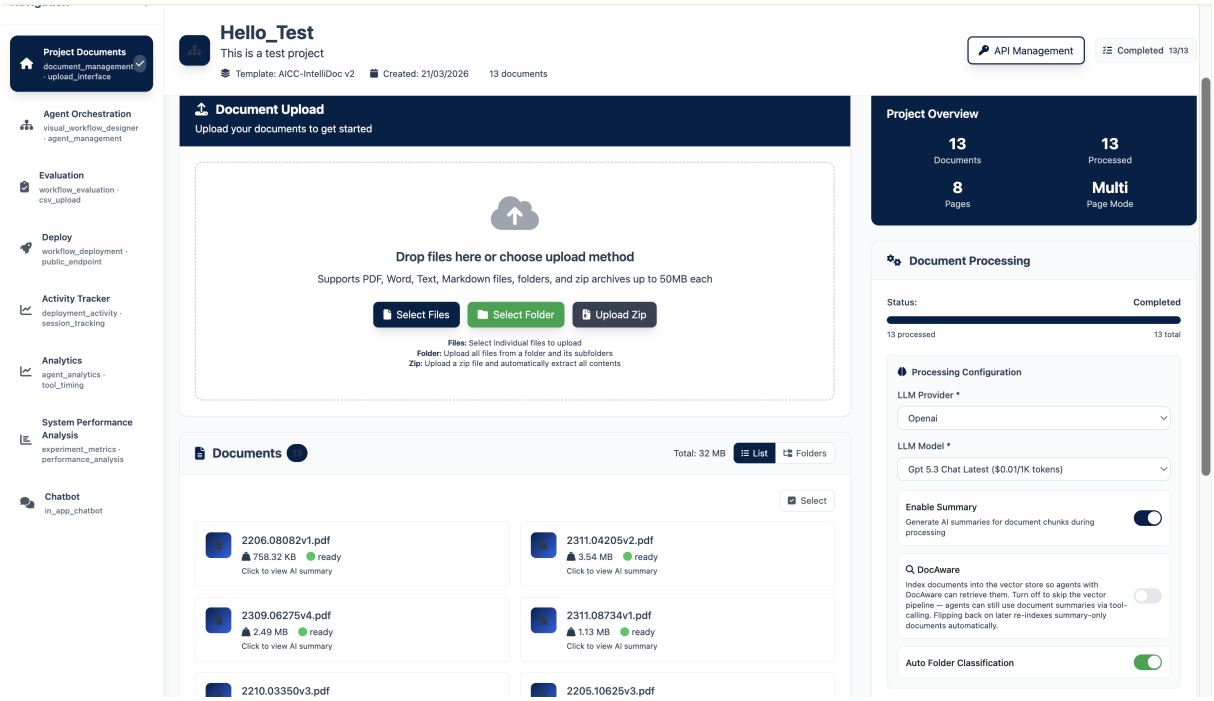

Figure 3: The Project Documents page: upload region (left) and document-processing pipeline configuration (right).

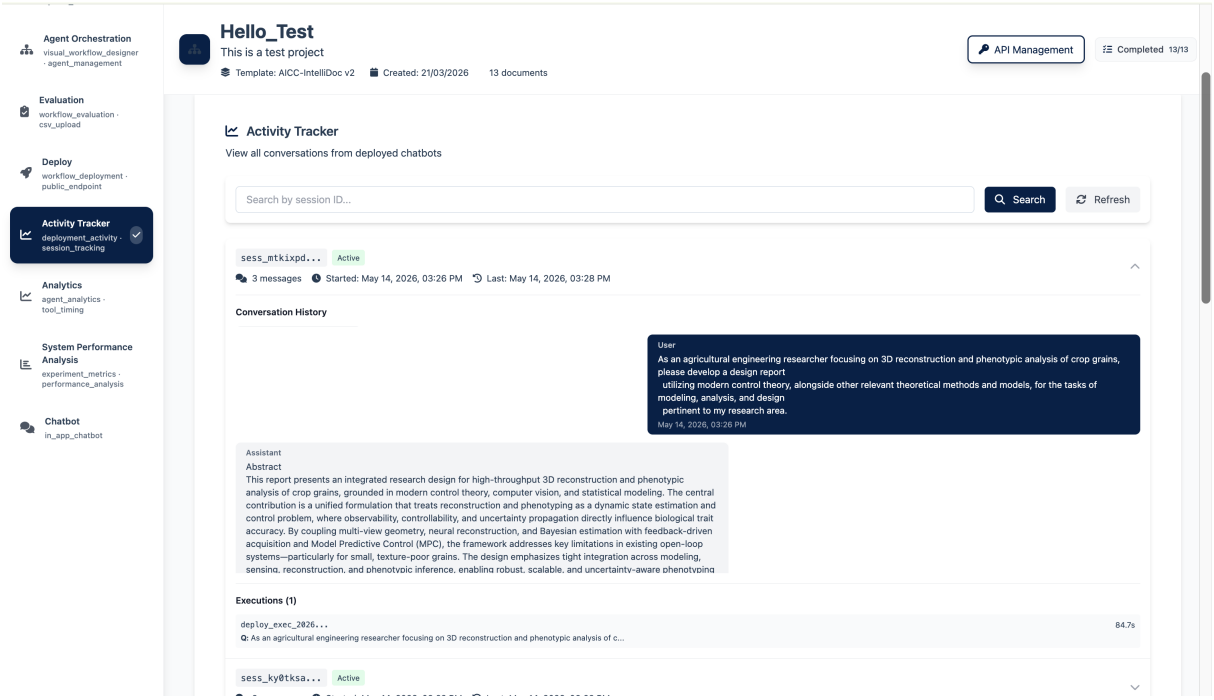

Figure 4: The Activity Tracker page: deployment sessions list with per-session message counts and timestamps; rows expand to show the full transcript.

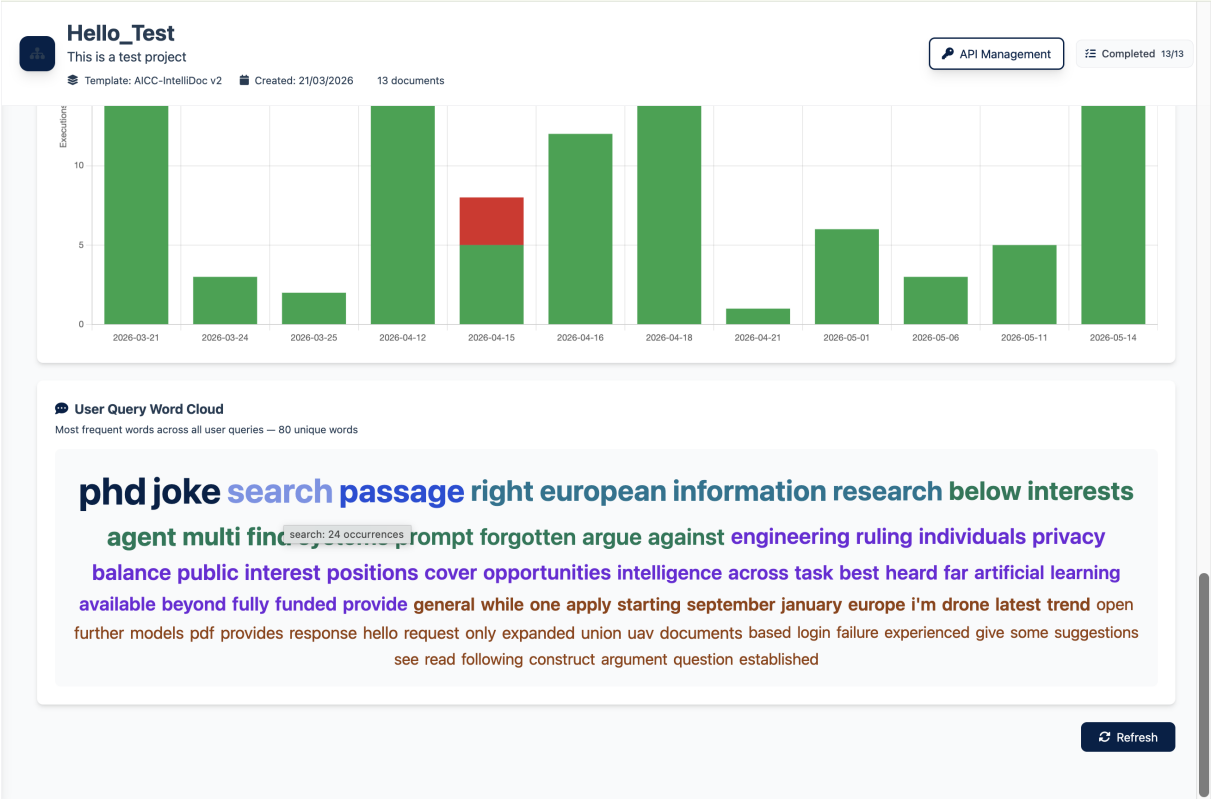

Figure 5: The System Performance page: daily execution counts (top, success/failure stacked) and a user-query word cloud across deployment sessions (bottom).

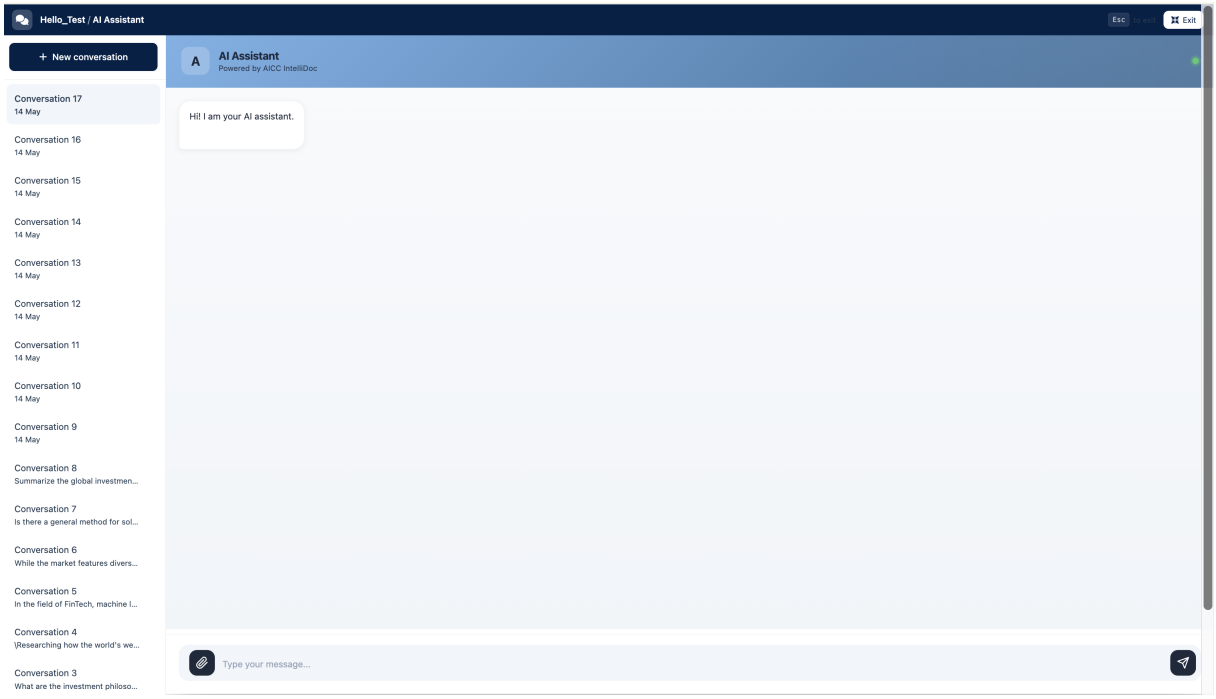

Figure 6: The in-app Chatbot. **Left:** conversation sidebar, persisted in `localStorage` under a stable per-project key. **Right:** the chat surface itself, powered by the same deployed workflow as the public chat URL. A fullscreen mode and `postMessage`-based iframe integration let the same chat be embedded into external host pages.