ChatGPT

Jaguar: AI Developer Agent for Workflow Automation

Overview

Jaguar is a codename for *The Spatial Network's* master AI developer agent – a conversational AI designed to help developers create and manage automation workflows using natural language. It combines **Open WebUI**, an extensible self-hosted AI platform [1], with **n8n**, an open-source workflow automation tool [2]. The goal is to provide functionality similar to proprietary platforms like Pipedream, but entirely open source and tightly integrated with AI. Jaguar's overarching mission is to **"guide the evolution of agents with natural wisdom,"** meaning it not only performs tasks on command, but also offers insightful guidance drawn from accumulated knowledge in the codebase and best practices. This agent runs in the cloud (hosted on DigitalOcean) with the following components:

- **Open WebUI** (hosted at `ai.thespatialnetwork.net`): Provides the chat interface and AI runtime for Jaguar, supporting multiple LLM models and tool integrations [3].
- **n8n** (hosted at `n8n.thespatialnetwork.net`): Acts as Jaguar's "hands," executing automation workflows. n8n is an extensible, self-hostable workflow engine that can **"connect anything to everything"** via its node-based interface [4]. Jaguar leverages n8n's public API and webhooks to create, read, update, and delete resources (workflows, credentials, etc.) on the fly.

By uniting a powerful natural language AI with a flexible automation backend, Jaguar allows developers to spin up integrations, agent behaviors, and workflow automations just by chatting with an AI assistant – effectively **an open-source alternative to Pipedream** that is enriched with AI-driven intelligence.

Target Users and Use Cases

Initial Users: Jaguar is initially intended for *The Spatial Network* developers – a team focused on building AI solutions for environmental regeneration and other mission-driven projects [5]. These users are technically proficient but seek to accelerate their workflow through AI assistance. Instead of manually configuring n8n workflows or writing boilerplate code, they can converse with Jaguar to do so.

Primary Use Cases:
- *Workflow Automation via Chat:* A developer describes an integration scenario in plain English (e.g., "Monitor a GitHub repo for new issues and post a summary to Slack daily"). Jaguar will interpret this request, create an appropriate n8n workflow (with trigger, nodes, and logic), and deploy it on the n8n server.
- *Continuous Workflow Improvement:* Jaguar can iteratively refine workflows. Developers can ask Jaguar to modify steps (e.g., "Add error handling to that workflow" or "Use a different API for this task"), and Jaguar will update the existing n8n workflow accordingly (update nodes, credentials, or logic as needed).
- *Credentials and Config Management:* Jaguar manages integration credentials in n8n as needed. If a workflow requires an API key or OAuth token, Jaguar has permission to create or update a credential in n8n's secure store, so the workflow can access external services seamlessly.

- *Code Repository Assistance:* For tasks that involve code generation or repository management, Jaguar can create or update GitHub repositories. For example, if an agent's logic requires a new microservice or function, Jaguar might generate a project scaffold or new script and use an n8n workflow (or direct API call) to commit this to a GitHub repo. This bridges the gap between high-level design and actual code implementation.

Long-Term Vision: As Jaguar matures, its use may extend beyond the internal dev team. It could be exposed (with proper access control) to other teams or even external collaborators who are not n8n experts. In those cases, Jaguar would act as a friendly *"AI DevOps assistant"* helping non-experts configure complex integrations with natural language – democratizing automation creation.

Goals and Objectives

The main goal of Jaguar is to **streamline the creation and evolution of AI-driven workflows and agents** by harnessing natural language and the wisdom embedded in existing knowledge bases. Key objectives include:

- **1. Accelerate Development:** Dramatically reduce the time and effort required to set up integrations or agent behaviors. Developers should be able to go from idea to a running workflow in minutes by conversing with Jaguar, rather than hours of manual configuration or coding. Jaguar leverages the fact that *"AI agents can do things like draft emails, book appointments... really anything you can dream of"* by tying into software APIs [6] – making those capabilities available on demand.
- **2. Leverage Open-Source Infrastructure:** Use and showcase open tools (Open WebUI, n8n, etc.) instead of closed platforms. n8n's commitment to visible source and self-hosting ensures we have full control and can extend the system freely [2]. This aligns with The Spatial Network's ethos of "OpenSourceEverything." Jaguar itself will be an evolving knowledge base in our repo that the AI can learn from and improve on over time, creating a virtuous cycle of open development.
- **3. Intelligent Guidance ("Natural Wisdom"):** Go beyond executing commands – Jaguar should provide guidance and best practices. For example, if a user asks for a certain workflow, Jaguar might suggest optimizations or caution against common pitfalls. This "wisdom" is drawn from the agent's training data, our internal repositories, and possibly principles inspired by natural systems (given our organization's focus on environmental regeneration). In essence, Jaguar serves as a mentor agent, not just an order-taker. It guides the evolution of other agents and workflows by injecting learned insights.
- **4. Dynamic Workflow Evolution:** Enable iterative improvement of automations. Jaguar should be capable of **self-refinement** loops: for instance, the user can set a goal ("handle 90% of incoming support inquiries automatically"), and Jaguar can attempt an initial workflow, test it (via simulated or real inputs), identify shortcomings, and refine the workflow. This addresses a scenario similar to one proposed by the community: using an AI to keep altering an n8n workflow until it meets the desired outcome [7]. Jaguar aims to make such continuous improvement possible in a controlled manner.
- **5. Secure and Manageable Deployment:** Ensure Jaguar operates within safe boundaries. It should have clearly defined permissions (CRUD on n8n workflows/credentials and limited repo access) and use them responsibly. All actions should be traceable for review. By achieving this, Jaguar can be trusted to handle more autonomy in the future.

Key Features and Requirements

1. Natural Language Interface (Conversational UX)

Jaguar provides a **chat-based interface** where developers converse with the agent much like they would with ChatGPT or a colleague. This interface is delivered through Open WebUI's frontend, offering a user-friendly chat experience in the browser [3]. Key requirements:

- **Multi-Turn Dialogue & Memory:** Jaguar should remember context within a conversation (using Open WebUI's conversation memory and history features) so developers can build complex requests iteratively. For example, after creating an initial workflow, the user could say "Now add an email notification to that workflow", and Jaguar understands it refers to the previously created workflow.
- **Clarity and Confirmation:** Jaguar will confirm interpretations and potentially show a summary of the workflow it intends to create or changes to apply, especially for destructive actions. For instance, if user says "delete workflow X", Jaguar might respond with "Are you sure you want to delete workflow X?" for safety.
- **Feedback and Explanations:** When Jaguar performs an action, it should explain what it did in simple terms. E.g., " I created a new workflow 'GitHub to Slack Notifier' with 3 nodes: a GitHub trigger, a filter, and a Slack node. I also saved your Slack webhook URL as a credential." This way, users learn from Jaguar's outputs, aligning with the goal of guiding users (and agents) with wisdom.

2. Integration with n8n for Workflow Control

At Jaguar's core is the integration with **n8n** – enabling the AI to actually perform operations in the automation server. This is achieved via n8n's REST API and webhooks, essentially treating n8n as a "backend" for the AI agent [8]. Requirements and design details:

- **Full CRUD on Workflows:** Jaguar can create new workflows, read (retrieve) existing workflow configurations, update them, and delete them on the n8n instance. This will likely use n8n's **Public API** endpoints (which accept and return workflow JSON). For creation and update, Jaguar will generate the JSON structure that defines the workflow nodes and connections. (Under the hood, Jaguar may employ templates or the logic from the AI Agents Masterclass examples for n8n integration workflows to construct these JSON definitions).
- **Credential Management:** Jaguar can create and update credentials in n8n via API, allowing newly created workflows to authenticate with external services. For example, if a workflow needs to call the GitHub API, Jaguar might automatically set up a "GitHub API Token" credential in n8n and reference it in the workflow. This requires that Jaguar holds a master API key or token for n8n with permission to manage credentials (hence **secure storage of n8n API credentials** is critical).
- **N8N Pipe Function:** Jaguar will utilize the **"n8n_pipe"** function (an Open WebUI pipeline plugin) to communicate with n8n [8]. This function acts as a bridge between the AI and a specific n8n webhook workflow. Key points:
- A dedicated Jaguar agent workflow will be set up in n8n, exposed via a webhook URL. The `n8n_pipe` plugin in Open WebUI is configured with that webhook URL (`n8n_url`) and an authentication token [9].
- When Jaguar needs to perform an action, it calls the `pipe` function, sending along the user's request or the specific command. The function will **POST the data to n8n** with the proper auth header and payload structure (including a session ID and the input text under a field like `chatInput`) [10].
- n8n receives this via a special workflow (let's call it "Jaguar Backend Workflow"). That workflow uses n8n's internal capabilities (e.g., nodes, code, or the n8n API itself via HTTP Request nodes) to carry out the command – such as creating a new workflow – and formulates a response. For consistency, we'll design the n8n workflow to output a JSON with a field (e.g., `output`) containing the result or answer text.

- The `n8n_pipe` function then captures the response and injects it back into Jaguar's conversation as the assistant's answer [11]. From the end-user's perspective, it feels like Jaguar directly answered with results (for example, "Your workflow has been created and is now active.").
- This mechanism allows asynchronous or heavy operations to be handled by n8n, while the LLM focuses on language understanding and decision-making. It essentially **offloads complex logic to a robust backend**, which is in line with Open WebUI's pipeline approach for heavy tasks [12].
- **Error Handling and Status:** Jaguar should handle cases where n8n might return an error (e.g., invalid workflow JSON, or network issues). Using the pipeline plugin, Jaguar can emit status messages – for instance, "⚠ There was an error creating the workflow: [error details]" – letting the user know something went wrong [13]. It will also log or display intermediate statuses (the `n8n_pipe` function is capable of showing a "Calling N8N workflow..." status indicator while the request is processing [14]). This keeps the user informed of progress during potentially long operations.
- **Example Scenario:** A developer says, "Jaguar, set up a daily email report of new signups." Jaguar (via LLM) decides a suitable workflow: a Cron trigger (daily) -> a database query or API call -> an email node. It then uses the `n8n_pipe` to call n8n, where a prepared Jaguar-backend workflow takes the request ("daily email report of new signups") and creates the actual n8n workflow with those nodes. Jaguar then replies in chat with something like, " I created the workflow 'Daily Signup Report'. It will run every day at 9 AM and email the report to the team." The developer can then ask Jaguar to show the workflow details or make adjustments if needed.

3. GitHub Repository Integration

Jaguar isn't limited to n8n workflows; it can also help manage code in GitHub repositories – for example, creating boilerplate for new agents or updating configuration files. This is facilitated through n8n (using GitHub integration nodes or HTTP requests to the GitHub API) or potentially through the OpenAPI Tool Server if we define a GitHub tool. Key requirements:
- **Creating Repositories or Files:** On command, Jaguar could initialize a new project repository (e.g., for a new microservice an agent might need). It would invoke an n8n workflow that uses the GitHub API (with stored credentials) to create a repo under the organization, then perhaps commit an initial README or code scaffolding.
- **Committing Changes:** For existing projects, Jaguar can commit new files or edit files. For example, "Jaguar, update the README of the `ai-open-agents` repo to include the new usage instructions." Jaguar can fetch the current README (through GitHub API), modify the content (possibly assisted by the LLM for the text generation), and then use an n8n GitHub node to push the update.
- **Pull Request Generation:** In cases where direct commits to main are undesirable, Jaguar could create a branch and open a Pull Request for review. This way, developers still retain oversight. Jaguar can tag the PR with a description of what it did.
- **Citing Knowledge Base:** Jaguar's knowledge of our repositories (like `ai-agents-masterclass`) means it can reference existing code or patterns. For instance, it might say "I created module X similar to the example in folder Y of the masterclass repo," ensuring transparency. This reuse of proven templates contributes to Jaguar's "natural wisdom" – it's learning from our collective codebase and reapplying that knowledge.

(Note: The exact mechanism for GitHub integration will be implemented via available n8n capabilities. n8n has nodes for GitHub that can create issues, releases, etc., and one can use a generic HTTP node for any GitHub API endpoint not covered by core nodes. Because n8n allows custom function nodes and API calls, Jaguar can

effectively instruct n8n to perform any Git operation, aligning with n8n's promise to let you "connect anything to everything" [4] *.)*

4. Knowledge Base and Learning Capabilities

Jaguar will be bootstrapped with a rich knowledge base to ground its responses and actions:
- **AI Agents Masterclass Templates:** We will incorporate patterns and code from the `ai-agents-masterclass` repository (which contains numerous example agents and integrations) as part of Jaguar's prompting or fine-tuning. This means Jaguar is aware of how certain agent workflows were built (e.g., LangChain integration, RAG agents, multi-step task managers) and can draw on those examples when needed. It provides Jaguar with *"reference implementations"* that embody best practices. Over time, Jaguar might even improve these patterns or suggest new ones, effectively **learning from and then extending** the masterclass knowledge.
- **OpenAPI Tool Server Utilities:** The custom tool server (`ai-open-agents` repository) provides additional actions (Supabase DB access, soon Nextcloud for files, etc.) [15] that Jaguar can use. Jaguar's design allows calling these tool APIs when necessary – for example, retrieving data from our Supabase database or publishing a WordPress article in the future. By having access to these tools (via OpenAPI integration in Open WebUI [16]), Jaguar's knowledge isn't just static text – it can actively query and use our data sources, making it more effective and context-aware.
- **Continuous Improvement:** Jaguar itself will be a project in our repository (with prompt templates, function definitions, etc.). As developers use Jaguar, we will gather feedback on its performance. Those insights, along with new code examples from our projects, will be used to refine Jaguar's underlying model or prompt strategies. In effect, Jaguar **learns from its own repo** – a practical step toward an AI that evolves. Success would mean each iteration of Jaguar becomes wiser and more capable, embodying an ever-growing corpus of "natural wisdom" from our team and domain.

5. Permissions, Security, and Ethical Guardrails

Because Jaguar can make impactful changes in our systems, we must enforce strict permissions and guidelines:
- **Scoped Access:** Jaguar's access tokens (for n8n, GitHub, etc.) will be scoped to only the resources it needs. For n8n, a dedicated API user or token will be used that can manage workflows and credentials, but nothing beyond that environment. For GitHub, a bot account or fine-scoped token will allow repository changes in designated orgs or repos. This minimizes risk if Jaguar ever goes off-script.
- **Approval Mechanisms:** Especially for destructive operations (like deleting a workflow or wiping data) and for major code pushes, Jaguar should seek explicit confirmation. This can be built into Jaguar's conversation logic (it will phrase such actions as questions to the user, requiring a yes/no confirmation). Additionally, we might configure n8n workflows to flag certain operations for manual approval by an admin before execution.
- **Logging and Audit Trails:** Every action Jaguar takes via n8n will be logged. n8n itself logs workflow executions; we will ensure the Jaguar-triggered workflows log the incoming request and outcome. Similarly, any GitHub commits by Jaguar can be traced to the bot account. These logs serve for both debugging and accountability.
- **Ethical Constraints:** Jaguar's guidance ("natural wisdom") should also encompass ethical considerations. For example, if asked to do something that violates data privacy or our environmental mission, Jaguar should politely refuse or seek human verification. We will bake in organizational values into Jaguar's prompt context (for instance, reminding it that our goal is aiding regeneration of Earth, so solutions that contradict

that are discouraged). While an AI agent cannot have true moral agency, it can be steered with these principles.
- **Fallback and Manual Override:** In cases where Jaguar is unsure or the action is beyond its current capability, it should be honest about it and possibly suggest involving a human or an alternative approach. The system should never leave a request hanging or make up a dangerous action. Our design will include a safe fallback where Jaguar says "I'm not confident how to proceed; perhaps we should handle this manually." This ensures that Jaguar's involvement remains a net positive and not a liability.

6. Performance and Scalability

Jaguar's architecture is cloud-based and modular to ensure it performs efficiently:
- **LLM Backend:** Open WebUI allows choosing different model backends (local or remote). Initially, we might use a moderately sized model (for example, a Llama-2 variant or OpenAI GPT-4 via API if allowed) to balance performance with capability. The server is configured with WebSocket support and, if available, GPU acceleration to handle the model inference [1] [17]. We expect sub-5-second response times for most simple queries, though complex multi-step workflow generation might take longer (due to calling n8n, waiting for responses, etc.).
- **Asynchronous Workflow Execution:** The use of the `n8n_pipe` ensures that Jaguar isn't blocking on long operations. n8n can execute the workflow in the background and respond when done, while Jaguar's UI can show a spinner or status updates. This decoupling improves perceived performance and keeps the UI responsive.
- **Scalability:** Both Open WebUI and n8n are horizontally scalable if needed. We can run multiple Open WebUI instances behind a load balancer if many concurrent users chat with Jaguar. n8n can also be scaled or its executions externalized if workflow volume grows. Given initial use is by a small dev team, our current single-instance deployments on DigitalOcean are sufficient. However, the design is cloud-agnostic – we could deploy Jaguar on other cloud infra or on-prem if needed (leveraging Docker and Kubernetes as described in our repos [18]).
- **Monitoring:** We will monitor resource usage (CPU/GPU, memory) of the Open WebUI and n8n servers. If Jaguar's usage spikes (e.g., someone triggers a very large workflow creation), we need alerts to ensure it doesn't overwhelm the droplet. Over time, usage patterns will inform if we need to allocate more resources or optimize the LLM's performance (via quantization, model distillation, etc.).

Technical Architecture Diagram

(for illustrative purposes – see textual description below)

Components:
- **Open WebUI Server (Jaguar Frontend):** Hosts the chat interface and LLM. Jaguar "lives" here as a conversational agent. It has the `n8n_pipe` function module installed (as shown in the Open WebUI community functions list [8]). The LLM processes user messages, and when it needs to perform an action, it invokes the `pipe` function with the appropriate parameters.
- **Jaguar Agent Workflow (in n8n):** A specific workflow on the n8n server designed to interface with Jaguar. It likely starts with a Webhook trigger (listening on a URL that matches the `n8n_pipe` configuration). When triggered, it reads the input (e.g., the user's request or question), and routes it to the correct n8n operations. For example, it might have a switch: if the input asks to create a workflow, the n8n workflow will then use the n8n API (through an HTTP Request node or n8n's internal function) to create a workflow. If the input is a general question or something Jaguar itself can handle, n8n might just route it back with an

answer. This workflow ends by returning a JSON payload with an `output` field containing the answer or result that Jaguar should present [10].
- **N8n Main System:** Apart from the Jaguar agent workflow, the n8n instance holds all user-created workflows and credentials. Jaguar essentially acts as an admin/operator on this system via API. The n8n instance is also integrated with external services via its hundreds of nodes. This means Jaguar doesn't need direct HTTP access to, say, Slack or Twilio – it can command n8n to use its Slack node or Twilio node to do something. The **range of integrations available to Jaguar is effectively the sum of all n8n's capabilities** (which include popular databases, SaaS APIs, email, etc. out of the box). This is powerful – Jaguar can orchestrate multi-step processes involving diverse systems simply by leveraging n8n's library of nodes.
- **OpenAPI Tool Server (Optional for extended tools):** In our deployment, we also run a custom FastAPI server (from `ai-open-agents`) exposing certain tools via OpenAPI [19]. Open WebUI can consume these as "tools" for the AI as well [16]. While not strictly required for Jaguar's core features (since n8n covers a lot), this tool server can handle specialized tasks (e.g., direct database queries via Supabase, file operations in cloud storage, etc.). Jaguar will use these as needed, expanding its abilities beyond what n8n can do out-of-the-box.
- **Data Flow:** When a user issues a command in Jaguar's chat, the LLM processes the intent. If the query is informational, Jaguar might answer from its knowledge. If action is needed, Jaguar formulates a structured request (often behind the scenes) and sends it through `n8n_pipe`. The request hits the Jaguar Agent Workflow in n8n, which then performs the necessary steps. Once done, the n8n workflow returns a result which flows back to Jaguar and is presented to the user. This round-trip is secure (via token authentication) and can be logged at each step for auditing.

(No single static diagram is provided here in text, but conceptually one can imagine the user on one side interacting with Jaguar (Open WebUI), Jaguar calling out to n8n (possibly illustrated by an arrow labeled "n8n_pipe webhook call"), and n8n executing a workflow then returning data back to Jaguar.)

Development Roadmap

Jaguar's implementation will proceed in phases to ensure a stable and useful product:

- **Phase 1: MVP (Minimum Viable Product)** – *Focus:* Basic workflow creation via chat.
- Set up Open WebUI and confirm it's stable on `ai.thespatialnetwork.net`. Install/configure the `n8n_pipe` function with our n8n endpoint.
- Create the initial Jaguar Agent Workflow in n8n that can handle at least one or two commands (perhaps "create workflow" and a simple "list workflows"). Test Jaguar end-to-end: e.g., user says "Create a workflow that sends me an email daily," Jaguar through n8n creates a dummy workflow, and responds.
- Implement basic credential creation through the agent (likely for a predefined service like an email SMTP credential, to demonstrate the concept).
- Ensure logging and error handling for these actions.

- **Success Criteria for Phase 1:** Jaguar can successfully take a simple English request and produce a functioning n8n workflow that performs the described task. The spatial network devs can use this without encountering major issues or needing to manually fix the workflow.

- **Phase 2: Expanded Capabilities** – *Focus:* Update & delete workflows, GitHub integration, knowledge base utilization.

- Add commands for updating workflows (e.g., “add a new node to X workflow” or “change the schedule of Y workflow”). This requires Jaguar to retrieve workflow JSON, modify it, and send an update via API.
- Add support for deleting or deactivating workflows by name or ID, with confirmation steps.
- Introduce the GitHub integration path: create at least one n8n workflow that Jaguar can trigger to perform a repo operation (perhaps creating an issue or committing a file). Let Jaguar use it when user asks for repo-related tasks.
- Integrate more of the knowledge base: e.g., allow Jaguar to answer questions about our AI architecture by referencing `ai-agents-masterclass` content (this could be done via a vector store and a RAG approach in Open WebUI). While not core to automation, it enhances Jaguar’s usefulness as a mentor.

- **Success Criteria for Phase 2:** Jaguar can handle a full CRUD cycle on workflows and at least one form of repository management. Users have successfully used Jaguar to modify an existing complex workflow (e.g., integrate a new API into an existing pipeline) just by instructions. Jaguar demonstrates use of stored knowledge by suggesting a solution referencing prior work (e.g., “We did something similar in Project X, I’ll follow that pattern.”).

- **Phase 3: User Experience & Wisdom** – *Focus:* Refine Jaguar’s conversation skills and guidance.

- Incorporate more sophisticated dialog management. Possibly implement a form of **tool selection reasoning**, where Jaguar internally decides whether to answer directly or call n8n, using a chain-of-thought prompt or a planner (this could draw on LangChain agents logic which we have examples of).
- Enhance the “natural wisdom” aspect: enrich Jaguar’s prompt with guidelines (like “if the user’s request can be solved in a simpler way, suggest that first” or “always consider the environmental impact if relevant”). Potentially integrate a **“reflection” step** where Jaguar double-checks the workflow it’s about to create against a checklist of best practices (we can maintain this checklist in the knowledge repo).
- UI/UX improvements: possibly develop a simple **workflow visualization preview** that Jaguar can send (for example, Jaguar could send a snapshot image or a textual diagram of the workflow nodes it created). This could be done by leveraging n8n’s ability to export a workflow image or using a text-based diagram.
- Gather feedback from the dev team and iterate on Jaguar’s responses to make them more clear, concise, and helpful.

- **Success Criteria for Phase 3:** Users report that interacting with Jaguar feels intuitive and that Jaguar often provides useful suggestions beyond just executing commands. There is a measurable reduction in time to build or fix workflows thanks to Jaguar’s proactive tips. Jaguar’s output quality (both in chat and in the artifacts it creates) is high enough that minimal corrections are needed.

- **Future Phases:**

- **Multi-Agent Collaboration:** Explore having Jaguar spawn or coordinate with other specialized agents. For example, Jaguar could delegate a sub-task to a “research agent” that looks up documentation, or to a “testing agent” that verifies a workflow’s execution. The masterclass content on swarm agents and evaluation frameworks [20] [21] might inform this.

- **External User Access:** If opening Jaguar to non-developers or a community, implement user accounts, rate limiting, and perhaps a sandbox mode where external users can experiment without affecting production systems.
- **Integration of Voice or Messaging Apps:** Allow interacting with Jaguar via voice (speech-to-text) or through Slack/Telegram by piping those inputs into the Open WebUI API. This would make Jaguar accessible in more contexts (imagine a project manager asking via Slack, "Jaguar, how many workflows failed last night?" and Jaguar (with n8n's data) responding).
- **Continuous Learning & Auto-Updates:** As our repositories (knowledge) grow, establish a pipeline for updating Jaguar's model or prompt. Possibly fine-tune a smaller LLM on internal data for more accurate domain-specific responses. Also, use n8n itself to schedule regular updates (like syncing new FAQ entries or project docs into Jaguar's knowledge store).

Success Metrics

To evaluate Jaguar's success, we will track several metrics and outcomes:

- **Adoption and Usage:** How frequently do the Spatial Network developers use Jaguar for their tasks? Our goal is that at least 80% of new internal automation workflows are initiated via Jaguar (instead of manually in n8n) once the tool is fully introduced. A rising usage trend would indicate Jaguar is providing value.
- **Efficiency Gains:** Measure the time taken to implement a workflow with Jaguar vs without. For example, if a typical integration took 2 hours to set up manually and Jaguar can do it in 15 minutes of conversation and tweaks, that's a huge win. We will collect anecdotes or logs of workflow creation sessions to estimate time saved.
- **Accuracy and Reliability:** Track the success rate of Jaguar-created workflows. How often do they run correctly on the first try? How many iterations are needed to fix errors? We aim for Jaguar's outputs to be correct or very close, requiring at most one minor edit for 70% of cases in the MVP stage, improving to 90% as the system learns.
- **User Satisfaction:** Gather qualitative feedback from the dev team. Are Jaguar's suggestions and explanations clear? Do they trust the agent with more complex tasks over time? A simple survey or regular meeting can capture this. We expect initial skepticism to turn into confidence as Jaguar proves its worth.
- **Reduction in Manual Effort:** Count how many workflows or scripts were **fully** generated by Jaguar. Each such instance is a concrete example of avoided manual coding/configuration. Also, count how many existing workflows were updated via Jaguar's help (showing it's useful not just for creation but maintenance).
- **Knowledge Utilization:** Monitor if Jaguar is successfully using the knowledge base. For instance, when asked questions about past projects or for recommendations, does Jaguar produce answers that reference our internal docs or prior solutions? If Jaguar starts to surface information from the masterclass repository or tool server docs in its answers (with correct context), that indicates the knowledge integration is working.

By observing these metrics, we will know if Jaguar is achieving its purpose. Ultimately, success is defined not just by *usage* but by Jaguar truly **augmenting our development process with intelligence and efficiency** – allowing the team to focus on high-level creative problem solving while the AI handles the repetitive or complex integration work.

Conclusion

Jaguar is an ambitious fusion of conversational AI and workflow automation, positioned as a guiding "master developer" agent for The Spatial Network. By standing on the shoulders of open-source giants – **Open WebUI's AI platform and n8n's automation engine** – Jaguar exemplifies our commitment to open, collaborative, and forward-thinking tool development. It aims to empower developers to build and evolve agents and workflows rapidly, all while embedding the collective wisdom of our domain (and the natural world's inspiration) into every solution.

Upon completion of this project, we expect to have a **boilerplate framework** (a template) that not only serves our needs but can be shared with the community as a reference implementation of an AI-assisted development environment. Jaguar will continuously evolve, but this PRD establishes the foundation and vision: a future where developing complex automations is as easy as having a conversation, and where each interaction with the AI makes the system smarter for everyone.

Sources: The design and requirements above draw on our existing AI platform documentation and the capabilities of the tools in use. Notable references include the Open WebUI documentation for integrating external tools [16] and pipeline functions like the n8n Pipe (which *"allows you to chat with an N8N AI Agent workflow within Open WebUI"* [8]), as well as n8n's open-source philosophy (*"visible source code, self-hostable, connect anything to everything"* [2]). These inform how Jaguar can securely and effectively bridge the gap between natural language intent and automated workflows. The *ai-open-agents* repo highlights the use of custom OpenAPI tool servers for giving AI agents real-world actions [16] [15] , an approach we leverage for Jaguar's extended capabilities. Finally, inspiration is taken from community discussions about AI modifying its own workflows [7] , reinforcing that Jaguar's vision is at the cutting edge of AI-agent integration. Jaguar will turn those discussions into a practical reality, one wise workflow at a time.

---

[1] [17] Home | Open WebUI
https://docs.openwebui.com/

[2] [4] [7] CustomGPT creating and modifying workflows by API to n8n : r/n8n
https://www.reddit.com/r/n8n/comments/1incdv8/customgpt_creating_and_modifying_workflows_by_api/

[3] [5] [15] [16] [18] [19] README.md
https://github.com/The-Spatial-Network/ai-open-agents/blob/03f939ba86443c9259e38e3b99beb3fc8702353d/README.md

[6] [20] [21] GitHub - The-Spatial-Network/ai-agents-masterclass: Follow along with my AI Agents Masterclass videos! All of the code I create and use in this series on YouTube will be here for you to use and even build on top of!
https://github.com/The-Spatial-Network/ai-agents-masterclass

[8] [9] [10] [11] [13] [14] N8N Pipe Function • Open WebUI Community
https://openwebui.com/f/coleam/n8n_pipe

[12] ⚡ Pipelines | Open WebUI
https://docs.openwebui.com/pipelines/