Workflow Spec: AI PM Ops Copilot

Version 1.0 • 2026-02-27

Overview

AI PM Ops Copilot is an agentic workflow built in n8n that converts a job description into structured requirements, retrieves grounded evidence from your resume/portfolio via a RAG service, computes a Fit/Gap matrix, and generates final application assets (ATS resume bullets, LinkedIn outreach, and interview questions) using a local Ollama model.

System diagram

Runtime components

n8n (Docker container): Orchestrates the workflow steps; typically accessed at localhost:5678.

Ollama (host or container): Local LLM server used for three generations: requirements JSON, Fit/Gap table, final assets.

RAG evidence API (host or container): Retrieves relevant chunks from resume/portfolio and returns evidence bullets plus citations.

Workflow steps

Node	Purpose	Key Inputs	Key Outputs	Notes
On form submission (n8n Form Trigger)	Collect inputs from a form	job_title, company, job_description, notes	JSON item (form fields)	Form Trigger cannot use Respond to Webhook; use Form Ending page for display.

HTTP Request → Ollama #1	Extract requirements	Prompt + JD (from form)	message.content (should be strict JSON)	Use format=json if supported; otherwise parse defensively.
Parse Requirements JSON (Code)	Validate/normalize JSON	raw text from Ollama	requirements object	Return error object when JSON is missing/invalid.
Evidence - RAG (HTTP Request)	Retrieve grounded evidence	Role + requirements lists	answer (bullets), citations[]	Citations should include source + chunk/page id.
Assemble FitGap Prompt (Code)	Compose Fit/Gap prompt	requirements + evidence + citations	prompt string	Keep strict rule: only claim using evidence.
FitGap - Ollama #2 (HTTP Request)	Generate Fit/Gap matrix	prompt string	message.content (markdown table)	Ensure request body is valid JSON; content references the prompt.
Assemble Final Prompt (Code)	Compose final outputs prompt	job meta + fitgap + evidence + citations	prompt string	Adds rules: no invention; cite every bullet.
Generate Assets - Ollama #3 (HTTP Request)	Generate final deliverables	final prompt	message.content (assets text)	Consider a hard max length to control verbosity.
Convert to File (Convert to Text File)	Turn text into binary file	Text field (e.g., text)	binary data (e.g., data)	Text Input Field must be a field name, not an expression.
n8n Form (Form Ending)	Show output after submission	assets text or link	Rendered ending page	Recommended for Form Trigger since webhook response is unsupported.

Interfaces and endpoints

From inside the n8n container, use host.docker.internal to reach services running on the host machine. Typical endpoints used in this project:

Ollama chat: `http://host.docker.internal:11434/api/chat`

RAG evidence: `http://host.docker.internal:8000/ask`

Data contracts

Requirements JSON schema (returned by Ollama #1)

```
{
"must_haves": ["..."],
"nice_to_haves": ["..."],
"responsibilities": ["..."],
"keywords": ["..."],
"seniority": "...",
"domain": "..."
}
```

RAG response shape (Evidence API)

```
{
"answer": "8-12 bullets...",
"citations": [
{"ref": "[1]", "source": "Resume (MAR).pdf", "chunk_index": 11}
]
}
```

Reliability and guardrails

• **Strict grounding:** downstream prompts must instruct 'use ONLY evidence; do not invent'.

• **Parse hardening:** if Ollama returns non-JSON, stop early with a clear error and display it on the Form Ending page.

• **Prompt wiring:** ensure every Ollama call uses the assembled prompt field (avoid empty messages).

• **Safe filenames:** avoid colons in filenames; prefer Date.now() or ISO with ':' replaced.

• **Observability:** log the raw model output and parsed object for each stage in n8n execution data.

Deployment notes

To persist outputs to your host machine, mount a host directory into the n8n container (for example host ~/n8n-outputs → container /outputs) and write to /outputs/.... Otherwise, keep the flow UI-first: show results on the Form Ending page and/or enable download from the Convert to File output.

Change log

v1.0 - Initial workflow spec covering Form Trigger limitations, Ollama + RAG integration, and output handling.