My approach to building the AI Resume Customization Agent today was based entirely on first-principles thinking. Before I wrote any code, I broke down the challenge into its most basic elements. I concluded that any agent must perform four key tasks in order: first, extract information accurately to turn unstructured resume and job description text into a format a machine can read; second, conduct a semantic analysis to spot gaps between a candidate's experience and an employer's needs; third, rework the content to match the job; and finally, format the output according to strict rules to make sure it is ATS-compliant.

This foundational analysis influenced every decision I made, especially choosing my development environment. The project had two options, but a risk assessment made the choice clear. Emergent.sh had a limited credit system that posed an unacceptable risk of failure. A simple bug could use up my resources and stop the project. Plus, its high level of abstraction went against the need to build the agent's logic from the ground up. So, I decided to use n8n. By using its open-source community edition with Docker on a free-tier cloud virtual machine, I eliminated risks related to costs and usage. This allowed for the unlimited iteration necessary for a high-quality outcome. More importantly, n8n's visual, node-based interface provided the best way to create the modular, auditable workflow I needed.

With the strategy and tools set, I started by planning out the architecture. Using Excalidraw, I drew both a high-level component diagram and a detailed workflow plan. This created a clear blueprint before I began implementation. This blueprint was then executed in n8n as a tangible, multi-stage process. The workflow starts with a Webhook trigger to receive raw text inputs. The first two stages involve separate HTTP requests to a large language model (LLM). One request parses the job description into structured keywords, while the other parses the resume into a predictable JSON schema. The core of the agent's intelligence is in the third stage: a loop that processes each work experience bullet point. Inside this loop, a precise LLM call rewrites each bullet to make it more impactful and aligned with the job description's context. Finally, once the loop is done, a Code node combines the rewritten content and creates the final ATS-optimized text document.

The outcome is not a single prompt trying to do everything at once, which would be fragile and hard to manage. Instead, it is a strong and clear system where each step reflects a fundamental truth about the problem. This project shows how starting with first principles leads to a more thoughtful, effective, and durable AI solution.

Y. Kaushal Kumar.