

CANDIDATE BACKGROUND Engineer (Temenos SWE) → MBA (IIT Kharagpur) → Enterprise BD, New Logos (Hexaware)	SYSTEM STATUS In Production (Daily 03:30 Run + Electron GUI)	MODELS deepseek-v4-flash and deepseek-flash (Routed by Task)
--	---	---

The Job Search Tailoring Engine

Six stages, four of them models, that draft evidence backed application packets and stop short of sending them.

Tailoring an evidence backed resume and cover letter by hand took me 45 to 60 minutes per role, and the work was repetitive and prone to layout slips. I built a multi-agent loop that ingests roles from five public ATS APIs, scores fit against a rubric, drafts the pack, gates it through a deterministic linter and a critic agent, compiles a strictly one page PDF, and surfaces only the roles that need me. Every claim it writes is drawn from the verified knowledge vault described in Case Study 01. It does not submit anything, and where exactly that line sits is the whole design.

6 Stages

Fetch to Publish

Four run on a prompt contract in pipeline/bots/. Fetch and Publish are deterministic code.

1 Page

Every Pack, Deterministically

Binary search zoom with a hard 0.85 floor. Below it the build fails rather than shrink text.

4 States

Reach the Human

Held, Revise, Ready, Published. Everything else runs unattended.

45 to 60 min

Per Role, Before This Existed

What a Ready pack leaves is reading two one page PDFs and sending, a few minutes. My own timing, not instrumented.

Pipeline: Task Based Model Routing

1. FETCH	2. SCOUT	3. ANALYST	4. WRITER	5. CRITIC	6. PUBLISHER
No LLM 5 ATS APIs raw/jobs/*.json	deepseek-v4-flash Extracts the JD jd.md	deepseek-v4-flash 6 criteria, 0 to 5 fit-brief.json	deepseek-flash Drafts the pack resume.md	deepseek-flash Pass or fail critic-verdict.json	No LLM Confirms the pack publish/

Those six stages are not six calls to the same model, and the split decides what the loop costs. Scout and Analyst parse and score against a fixed rubric, which is classification rather than persuasion, so they run on the cheaper model and keep high volume board sweeps affordable. Writer and Critic produce and judge the exact text a recruiter reads, so they get the stronger model at high effort. A weak Writer is a false economy: it fails the Critic gate and burns revise cycles, and the contract allows only two. Lint has already settled everything mechanical by then, so a Critic fail is a judgement one: an invented metric, a title the vault does not earn, or an evidence map row citing a wiki page that does not carry the claim. Two revise cycles are allowed, and on a third the role stops and waits for me.

State Is the File System

Six stages need somewhere to record which stage a role reached. There is no database: stage is derived from which files exist in grok/drafts/<slug>/. A folder holding jd.md but no fit-brief.json is waiting on the Analyst. One holding a go verdict but no resume.md is waiting on the Writer. A hold verdict with no resume is waiting on me.

Why it earns its place: recovery from a failed agent is git restore or deleting one file. No orphaned locks, no migrations, no cache to invalidate, and the whole state of the system is readable with ls.

A Real Agentic Deadlock

Deriving state from files while denying agents the tools to move files failed in production. The Critic wrote critic-revise.md and set the verdict to REVISE. The Writer picked up the folder, applied the revisions, and saved a new resume. Then nothing happened, on every run, forever.

Neither agent had shell tools, by design, so neither could delete the revise trigger. It stayed, and the Critic never re-judged.

Fix: clear_revise.py runs as a deterministic step in the loop. If the Writer wrote revise-applied.md after the verdict, it archives the trigger and drops the verdict so the Critic re-judges. The lesson generalised: when agents are denied tools, something outside them must own the state transitions.

What I Deliberately Did Not Build

Which returns to where the line sits. Three things this system could do, does not, and will not.

- No auto-apply bot.** Zero applications auto submitted. The system stops at the PDF. Submitting is manual, which keeps me accountable for every application that carries my name and keeps the system clear of platform bans.
- No scraping of protected platforms.** board-platforms.toml tests every platform on terms, documented public API, and obligations. Greenhouse, Lever, Ashby, SmartRecruiters and Workable pass. LinkedIn prohibits automated crawling, so it is never swept, searched or listed. A job link I paste into the dashboard is read once from the public guest page, and nothing else touches it. Workday was dropped because its jobs endpoint is undocumented and its terms could not be read in full.
- No raw log stream on the default screen.** Thousands of lines of model thinking are real but unreadable, and users go log blind. The opening view shows only the roles awaiting a decision.