Automated Job Search & Cover Letter Generator with Match Scoring (n8n + Gemini)

# Problem Explanation

Job searching is time-consuming and repetitive. Manually browsing job boards, evaluating postings, and drafting cover letters requires significant effort daily. The challenge was to automate this workflow so that a candidate receives top job matches, complete with scores and tailored cover letters, directly in their inbox.

# n8n Workflow Breakdown (Node-by-Node)

# A screen shot of a schedule trigger AI-generated content may be incorrect.

* Schedule Trigger – runs workflow daily.

A black square with red and green lines and a red symbol

AI-generated content may be incorrect.

* RSS Read – fetches latest job postings from job boards.

A computer screen shot of a code

AI-generated content may be incorrect.

* Function Node – formats and filters the job data.

A computer screen shot of a computer

AI-generated content may be incorrect.

* HTTP Request – sends job data to FastAPI for AI analysis.
* A screenshot of a computer

  AI-generated content may be incorrect.
* Message Model (Gemini) – generates JSON with match score and cover letter.

A computer screen shot of a code

AI-generated content may be incorrect.

* Code Nodes – parse and clean JSON responses.

A computer screen shot of a computer

AI-generated content may be incorrect.

* Google Sheets Node – logs job details, scores, and cover letters.

A screen shot of a message

AI-generated content may be incorrect.

* Email Node – sends daily summary email with top matches.

# API & Prompt Usage Explanation

FastAPI endpoints were built to interact with Gemini AI. Key endpoints:

**SCORE-JOB**: Takes resume and job description, returns a match score (1-100).

CODE

your role is System, you're in system message mode, you're an intelligent bot rating how closely a job listing is to a candidate skill set, on a 5 score**.{Prompt 1 Role Model}**

here's the job listing details:

{{ $('HTTP Request').item.json.data }}

here is sample resume :

{

"name": "Suraj Kumar",

"email": "cocsamcoc69@gmail.com",

"phone": "8340616431",

"skills": ["Python", "SQL", "Tableau", "Excel", "Power BI", "Pandas", "Numpy", "Matplotlib", "Seaborn", "BeautifulSoup"],

"education": "B.Tech Science and Technology, SRM Institute (2019-2023), CGPA: 8.12",

"experience": "Data Science Intern at AlmaBetter (2024-Present), Data Analytics Trainee at Trainity (Dec 2023 - Mar 2024)",

"location": "India",

"profiles": {

"github": "https://github.com/surajsamm",

"linkedin": "https://www.linkedin.com/in/suraj-kumar-7581b6355/",

"hackerrank": "https://www.hackerrank.com/profile/cocsamcoc69"

}

}**{Prompt 2 Role User}**

Read the resume carefully and output strictly in JSON format:

{

"name": "",

"rating": "/100"//out of 100

}**{Prompt 3 Role Model}**

**GENERATE-COVER-LETTER:** Takes resume and job details, returns a personalized cover letter.

you are intelligent bot perfect at creating cover letters for a job|[ Please take the candidates resume and create a customized cover letter to the job **{Prompt 1 Role Model}**

here is a job listing deatails:

Title:{{ $('RSS Read').item.json.title }}

Discription:{{ $('Code').item.json.job\_description }}

Name:{{ $('Code').item.json.company\_name }}

Location:{{ $('Code').item.json.location }}

Your Name:{{ $json.name }} **{Prompt 2 Role user}**

You are an intelligent assistant that generates professional and customized cover letters. and output strictly in JSON format:

{

"cover\_letter": ""

} **{Prompt 3 Role Model}**

Prompts are carefully structured to ensure outputs are always in JSON format for n8n parsing.

// Input: raw string JSON inside item

// Output: clean object (no nesting)

return items.map(item => {

// Parse the nested text field

const rawText = item.json.content.parts[0].text;

const parsed = JSON.parse(rawText);

return { json: parsed };

});

# Challenges and Solutions

* API Rate Limits – Solved by batching requests and caching repeated queries.
* Text Parsing – Solved with Function nodes to clean newline characters and validate JSON.
* Email Formatting – Solved by switching to HTML emails and replacing newlines with line break tags.
* Random Job Selection – Solved using Function node logic to pick random or sequential jobs.

# Summary of Learnings

This project demonstrated how n8n can orchestrate complex workflows by combining APIs, AI models, and data storage. I learned how to integrate Gemini for structured JSON outputs, handle data cleaning in n8n, and automate daily reports through email and Google Sheets. The biggest takeaway was how automation can drastically reduce repetitive work, making job applications faster, more targeted, and more effective.

Name Suraj Kumar

Email:Cocsamcoc69@gmail.com