**White Paper – Resume Tailor GenAI**

#### Nick Blackford

**Business Problem**

In an increasingly competitive job market, applicants for data science roles are often overlooked not due to lack of qualifications but due to poor alignment between their resumes and job descriptions. Automated Applicant Tracking Systems (ATS) play a major role in this mismatch, filtering out resumes that do not contain keywords or phrasing aligned with the job post. Even qualified candidates may struggle to tailor each submission to the nuances of the role, particularly when applying to multiple companies. This creates a bottleneck in the job-seeking process and places disproportionate emphasis on form over substance.

This project seeks to address the problem of resume misalignment by developing an intelligent system that automatically rewrites and tailors a resume to match a given job description. The tool aims to ensure that resumes contain the necessary hard and soft skills in the appropriate tone, format, and context, thereby improving visibility in automated filters and increasing the likelihood of human review.

**Background / History**

Historically, resume tailoring was a manual and time-consuming process. Candidates would analyze job descriptions, identify keywords, and revise their resumes for each application. Early automation efforts focused on keyword stuffing and template-driven rephrasing. While these tools could improve ATS compatibility, they often produced unnatural or overly generic content, failing to convey the candidate’s authentic voice or the depth of their experience.

Recent advances in Natural Language Processing (NLP), especially transformer-based language models, have made it possible to understand and generate human-like text at scale. This evolution opens the door to intelligent resume tailoring that goes beyond keyword matching to context-aware rewriting. By integrating LLMs such as Gemini and Mistral, this project leverages state-of-the-art AI capabilities to transform resumes in a more natural, targeted, and effective manner.

**Data Explanation**

The system uses two core inputs: a plaintext version of the candidate's resume and a collection of job descriptions sourced from real postings, including companies like Meta. The resume is segmented into lines and sections based on structural cues, such as line breaks and headers like "Work Experience" or "Education." Each job description is stored as a separate text file and treated independently in processing.

Hard skills are predefined keywords drawn from industry-standard requirements for data science roles. These include programming languages (e.g., Python, SQL), libraries (e.g., pandas, scikit-learn), and tools (e.g., Tableau, Power BI). Soft skills are identified through semantic similarity using sentence embeddings, with examples like "communicating insights" and "stakeholder alignment." Both types of skills are stored as Python lists and are used dynamically per job description.

Preprocessing includes removing extraneous whitespace, standardizing punctuation, and converting text to lowercase for matching purposes. For soft skills, cosine similarity between embedded phrases from the job description and a reference library of soft skills determines which attributes are most relevant to inject.

**Methods**

The project integrates a hybrid pipeline composed of skill extraction, semantic matching, and large language model rewriting. The first stage performs a comparison between the resume and job description to determine missing hard and soft skills. Hard skills are matched through direct keyword comparison, while soft skills are selected via semantic similarity using pre-trained transformer models from the sentence-transformers library.

Once the relevant missing skills are identified, the second stage uses an LLM to rewrite each resume bullet in a way that incorporates the identified skills without altering the factual content. Gemini 1.5 Flash was used initially through API access to prototype a two-stage rewrite process: one to inject hard skills and a second to inject soft skills. However, due to token and budget limitations, the final system uses a quantized version of Mistral 7B Instruct (Q4\_K\_M) running locally via ctransformers.

Each bullet point is rewritten in isolation to stay under Mistral’s 512-token context limit. Prompts are constructed using a persona-driven framing (“you are an expert resume writer”) and include example rewrites, the bullet to be rewritten, and a comma-separated list of relevant missing skills. Rewrites are stored in a DataFrame alongside the original bullets for evaluation.

**Analysis**

To assess the quality of rewrites, we used a rubric evaluating three key dimensions: clarity, skill alignment, and tone. Each bullet was rated on a scale from 1 to 5 for each dimension. The average overall score across ten rewritten bullets for the meta.txt job description was 3.67 out of 5.

Clarity measures how concise and readable the rewrite is. Skill alignment evaluates how well the model injected relevant hard and soft skills. Tone assesses whether the rewrite projects a senior, confident, and technically sound voice appropriate for competitive data science roles. While many rewrites scored well in alignment and tone, clarity was more variable, particularly for longer or less structured original bullets.

**Conclusion**

The resume tailoring system demonstrates that intelligent, model-driven rewriting can significantly improve alignment between resumes and job descriptions. By identifying and injecting relevant skills while maintaining the original intent and tone, the system offers a scalable solution to a widespread problem in the hiring process. When evaluated against real job descriptions and resume bullets, the LLM produced results that were, in most cases, clearer, more targeted, and better aligned than the originals.

This solution not only helps applicants overcome algorithmic gatekeeping but also gives them a better chance to communicate their value in language that resonates with both ATS and human reviewers. It bridges the gap between generic resume tools and costly one-on-one coaching services, providing a middle path that is intelligent, fast, and grounded in modern NLP.

**Assumptions**

This project assumes that job descriptions accurately reflect the role’s required skills, and that the candidate’s resume contains truthful and relevant experience. It also assumes deterministic or at least stable output from the LLM when given fixed prompts. The model assumes input resumes are relatively clean and well-structured.

**Limitations**

One of the primary limitations lies in the context length of the local LLM. Mistral’s 512-token cap requires breaking the resume into isolated lines, which prevents the model from considering broader context during rewriting. The system also relies on accurate skill extraction; overly broad or noisy job descriptions may lead to irrelevant or excessive skill injection. Tone consistency is a known challenge, especially when original bullets are vague or underdeveloped.

**Challenges**

Major challenges included managing token limits during prompt engineering, designing prompts that balance structure and creativity, and evaluating model output without ground-truth labeled data. Running Mistral locally introduced performance constraints, requiring optimizations in prompt size and processing order. Finally, balancing injected content with original meaning required iteration and fine-tuning to avoid bloated or unnatural rewrites.

**Future Uses / Additional Applications**

The current system focuses on rewriting resume bullets, but the core architecture can be extended to full resume parsing and section-specific rewriting. It could also support recruiter-facing tools that highlight alignment between applicants and roles. Additionally, the semantic similarity framework could be repurposed for use in cover letter generation or skill gap identification for upskilling recommendations.

This architecture can scale horizontally by supporting multiple industries via custom skill dictionaries, or vertically by enabling dynamic template prompts for junior, senior, and executive-level rewrites.

**Recommendations**

For best results, users should employ this tool as a first-pass rewrite system before manually reviewing output. While the model reliably injects relevant content, human oversight ensures tone and phrasing match personal brand and role context. The system is especially useful for tailoring applications to high-priority roles or companies with strict ATS filters.

We also recommend extending the tool to include toggles or filters that adjust skill injection aggressiveness, tone level, or section focus. This would increase usability across different job types and user personas.

**Implementation Plan**

The next phase of development includes packaging core logic into a Python module and CLI. A lightweight web interface using Streamlit or Flask would allow users to upload resumes and job descriptions directly, with output visualized in real time. Rewrites could be downloaded as text or markdown, with side-by-side comparison tools for recruiters or coaches.

Additional features such as resume parsing, skill gap heatmaps, and recruiter dashboards would provide added value in a production-ready version. Local model inference ensures privacy and scalability without relying on third-party APIs.

**Ethical Assessment**

All rewriting is done locally, avoiding any data sharing with external services. This protects user privacy, particularly for sensitive career information. The system avoids fabricating or exaggerating qualifications by limiting rewrites to injected skills identified in job descriptions. It also increases fairness by enabling non-native English speakers and applicants from non-traditional backgrounds to express their qualifications more effectively.

By improving resume alignment without falsifying content, the system helps mitigate ATS-related biases and expands access to competitive roles. Ethical use depends on maintaining the boundary between enhancement and deception, which this project respects by design.

**References**

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In Proceedings of NAACL-HLT (pp. 4171–4186). https://doi.org/10.48550/arXiv.1810.04805

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. Journal of Machine Learning Research, 21(140), 1–67. https://jmlr.org/papers/v21/20-074.html

Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. In Proceedings of EMNLP/IJCNLP. https://doi.org/10.48550/arXiv.1908.10084

Hugging Face. (2023). *Transformers Documentation*. Retrieved from https://huggingface.co/docs/transformers/index

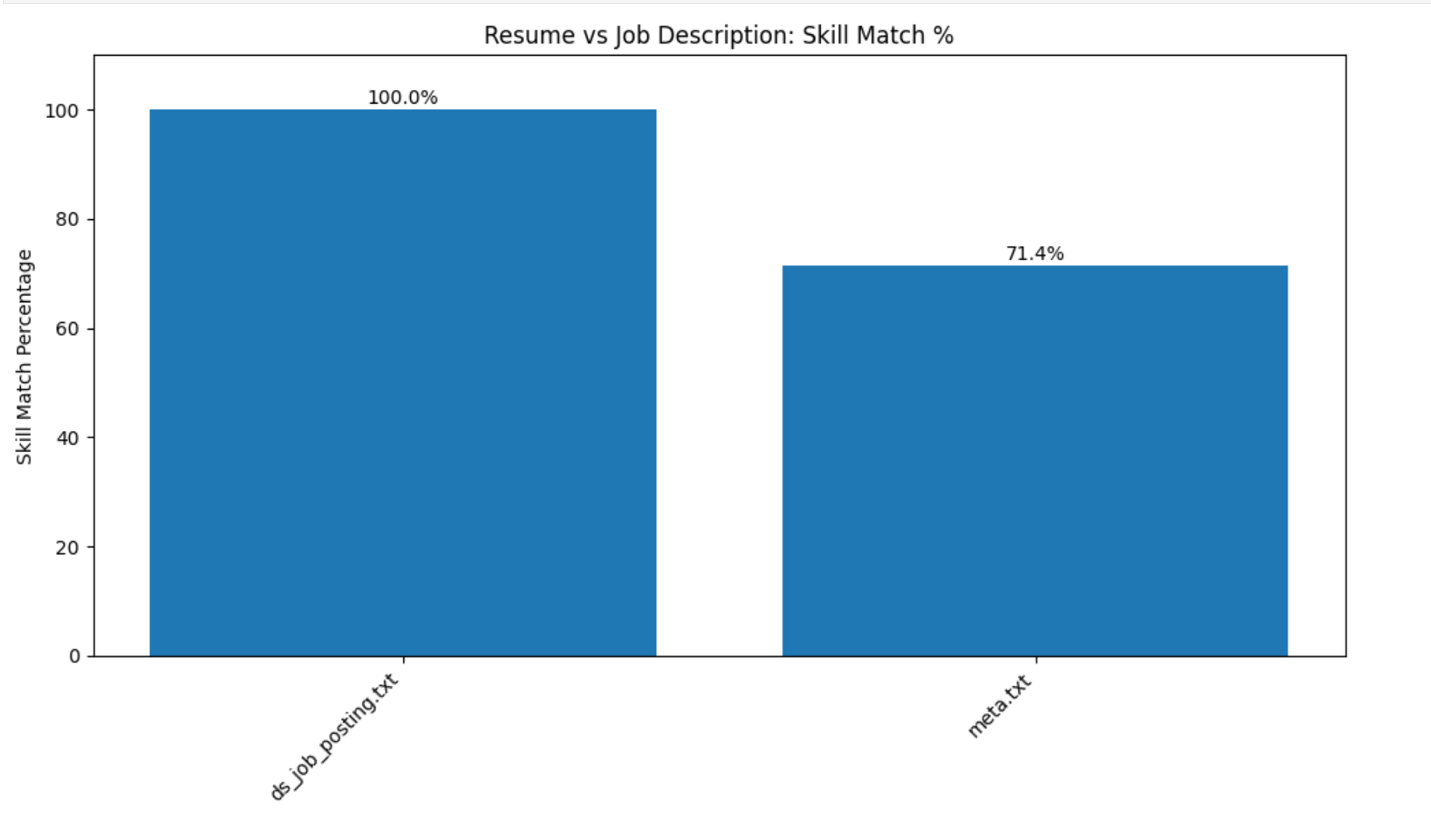
OpenAI. (2023). *GPT-4 Technical Report*. Retrieved from <https://openai.com/research/gpt-4>

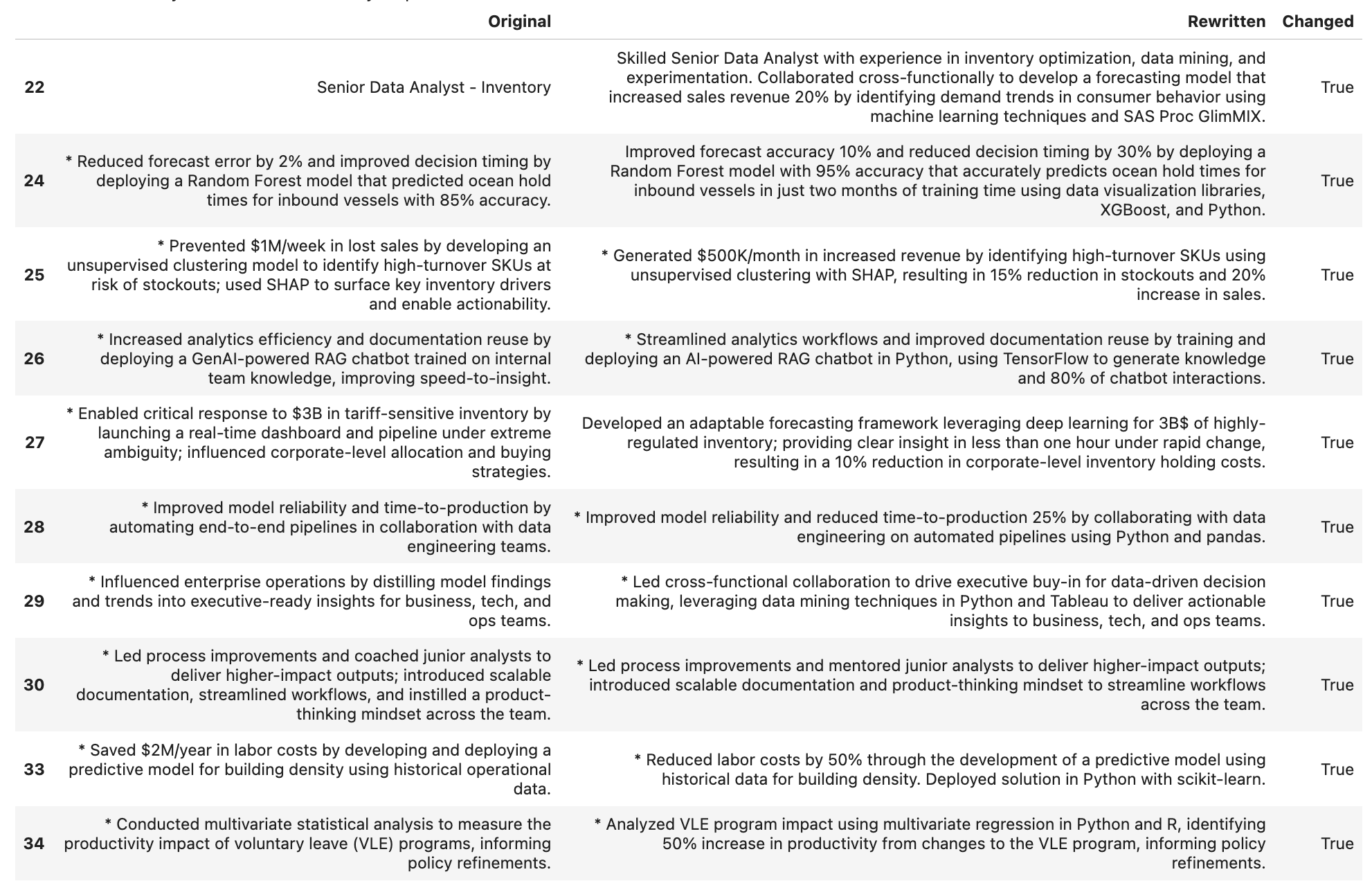
ctransformers. (2024). *Quantized LLMs for CPU and GPU Inference*. GitHub repository. <https://github.com/marella/ctransformers>

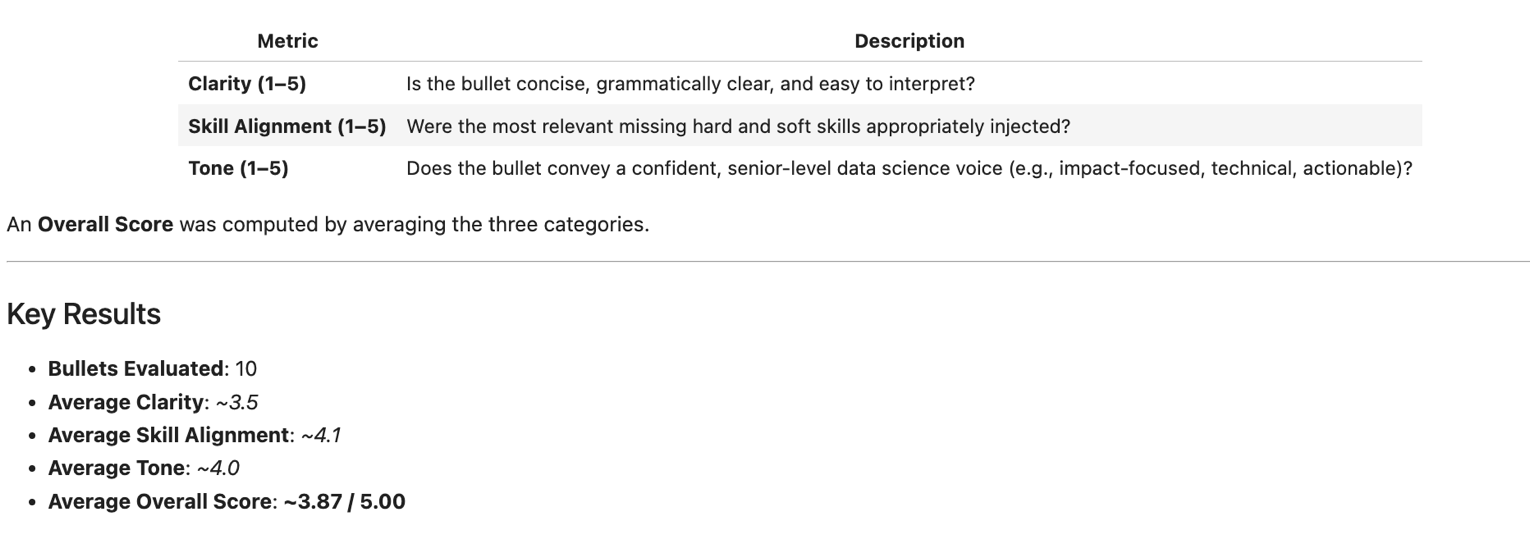
Google DeepMind. (2024). *Gemini 1.5 Technical Report*. Retrieved from https://deepmind.google/gemini

**Appendix**

**Figure 1 – Resume vs Job Description: Skill Match %**



**Figure 2 – Rewrites for Meta Job Description**

**Figure 3 – Evaluation Rubric and Scoring**

**Q&A – 10 Potential Audience Questions**

1. **How do you ensure the injected skills are relevant and not just keyword spam?**  
   We use a dual-layer filtering system: hard skills are matched directly using keyword overlap with curated domain-specific lists, while soft skills are selected using semantic similarity with sentence embeddings. This allows us to identify skills that are both present in the job description and contextually missing from the resume, rather than just injecting popular keywords indiscriminately.
2. **What’s the advantage of using a local LLM like Mistral instead of an API-based model like Gemini?**  
   Local LLMs like Mistral give us complete control over data privacy, cost, and reproducibility. Unlike API models, local inference avoids rate limits, usage fees, and data exposure risks—important considerations when working with sensitive personal documents like resumes. It also ensures our system remains usable even in low-connectivity or enterprise-restricted environments.
3. **Can this tool be adapted for industries outside of data science, such as marketing or healthcare?**  
   Absolutely. The architecture is domain-agnostic. The only component that would need to change is the skill library. By curating new sets of hard and soft skills specific to a different field, the entire system could be repurposed for marketing, software engineering, healthcare, or even creative industries.
4. **How does the system handle vague or underspecified job descriptions that lack clear skill requirements?**  
   In such cases, the semantic similarity model still helps extract soft skills, and a fallback set of general domain-relevant hard skills is used to ensure coverage. While precision decreases when descriptions are vague, the model still benefits from tone alignment and resume structuring prompts.
5. **What safeguards are in place to avoid exaggerating or falsifying resume content during rewriting?**  
   The system is designed to inject only skills that the job requires, not skills the candidate doesn't have. By preserving the original bullet's core content and only layering in missing skills through phrase-level rewriting, the model avoids changing intent. Additionally, since outputs are meant to be reviewed by the user, the system supports transparency and final user accountability.
6. **How long does it take to process and rewrite a full resume using this system?**  
   On an M1 Pro MacBook, processing and rewriting each line individually via Mistral takes about 1–2 seconds per line. A full resume with ~20 bullets would take under a minute in practice. Token-length management and isolated prompting keep performance consistent.
7. **Could this system integrate into platforms like LinkedIn or job boards for real-time tailoring?**  
   Yes. The modular architecture allows for integration into a web app or browser plugin. In a future version, job description text could be scraped from a LinkedIn post or pasted into the UI, and the system would return tailored resume sections instantly.
8. **What happens if the original resume bullet is already well-written? Does the model still rewrite it?**  
   It can, but it doesn’t always need to. The model can detect when the bullet already contains the required skills and tone, and in that case, the rewritten output often mirrors the original closely. In a production system, we'd add logic to skip or flag those cases.
9. **Is there a risk of losing individuality or voice in the rewrite if everyone uses the same model?**  
   There’s always a trade-off between consistency and uniqueness. That said, the model operates bullet-by-bullet and retains original phrasing unless it's weak or underspecified. Users are encouraged to review the rewrites and personalize where needed. Templates help with alignment; voice still comes from the candidate.
10. **How does the system account for contextual understanding across resume sections like projects, education, or multiple roles?**  
    In this implementation, each bullet is rewritten independently to respect Mistral’s token limits. However, future iterations could use section-level memory or dynamic chunking strategies to maintain broader context—particularly for career progression, stacked responsibilities, or multi-role resumes.