**International Journal of Scientific Research in Engineering & Technology**
Volume6, Issue6 (November-December 2026), PP: 35-43.
https://www.doi.org/10.59256/ijsreat.20260606006
www.ijsreat.com
ISSN No: 2583-1240

# Role Radar: An AI-Based Resume Intelligence and Skill Gap Analysis System

**Chaitra K J[1], Shreyas P Katte[2], Vinay[3], Vikas K V[4], Vishruth K V[5]**
[1]*Assistant Professor, Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology, Davangere, Karnataka, India.*
[2,3,4,5]*Department of Information Science and Engineering, Bapuji Institute of Engineering and Technology, Davangere, Karnataka, India.*

***To Cite this Article:*** *Chaitra K J[1], Shreyas P Katte[2], Vinay[3], Vikas K V[4], Vishruth K V[5], "Role Radar: An AI-Based Resume Intelligence And Skill Gap Analysis System", International Journal of Scientific Research in Engineering & Technology, Volume 06, Issue 06, November-December 2026, PP:35-43.*

Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution License; Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

**Abstract:** *Job seekers face two linked problems: finding postings that are genuinely open and suitable, and presenting their experience to automated screening without exaggeration. General-purpose language-model resume writers aggravate the second problem because they can invent skills, metrics, and seniority. This paper presents RoleRadar, a full-stack career platform built on one rule: deterministic code decides what is true, and the language model only rephrases it. A geometry-based parser turns an uploaded resume into atomic, provenance-tagged evidence units. Job requirements are mapped to that evidence on an eight-level support scale, and a category-aware six-factor model yields an explainable match score and a readiness verdict. A rule-based Truth Guard compares every tailored bullet with its source and reverts fabricated technologies, unsupported metrics, inflated titles, and broken sentences before one-page export. Supporting modules provide a taxonomy of 119 roles, prerequisite-ordered learning roadmaps, verified job ingestion from direct employer feeds, and an assistant restricted to the signed-in user's data. A regression suite of 1,268 test cases, internal validation harnesses, and an end-to-end acceptance run indicate that the pipeline is consistent and auditable. The system has not yet been evaluated with recruiters or in a controlled user study.*

**Keywords**: *resume tailoring, evidence grounding, applicant tracking system, job matching, hallucination control, skill gap analysis, large language models.*

## I. INTRODUCTION

Online recruitment has made openings easy to publish and discover, but it has also produced an information explosion in which candidates and recruiters must find a good fit among very large volumes of text [1]. Applications are commonly routed through applicant tracking systems (ATS) that parse a document before a person reads it, so a resume must be readable by software and honest about the candidate. These pressures are acute for students and early-career applicants in India, whose experience is often limited to coursework, projects, and internships.

Three problems recur in current tools. First, discovery is noisy: listings are stale, duplicated across sources, or redirect to aggregator pages instead of the employer's own requisition. Second, tailoring is risky: large language models (LLMs) readily produce fluent text that the source does not support, a failure mode known as hallucination [2]. In a resume, a hallucinated tool, metric, or job title is a misrepresentation that the candidate may have to defend in an interview. Third, scoring is opaque: a study of vendors of algorithmic hiring assessments begins from the observation that little is known about how such tools are built, validated, and examined for bias [3].

This paper describes RoleRadar, a full-stack platform that addresses all three problems with a single design rule: facts are established and verified by deterministic code, and the LLM is confined to rewording what that code has already established. The contributions are as follows.

- An evidence-unit representation that keeps provenance for every resume claim, and an eight-level mapping from job requirements to that evidence.
- A category-aware, six-factor match score with published weights, per-factor explanations, and a readiness verdict.
- A Truth Guard that checks every rewritten bullet against its source, reverts failures, and enforces conservation of evidence units.
- A verified-opportunity pipeline with URL classification, a lifecycle state machine, deduplication, and truthful compensation extraction.
- A role taxonomy and a prerequisite-ordered roadmap generator that turn skill gaps into a study plan without inventing skills.

Section II reviews related work, Section III presents the architecture and implementation, and Section IV the methods. SectionV reports validation results, Section VI discusses limitations, and Section VII concludes.

## II. PROLEM STATEMENT

Job seekers lack a unified and transparent method to determine how well their verified qualifications match a job description, identify the skills they are missing, and tailor their resumes without adding unsupported claims. Existing workflows can expose candidates to stale or duplicated vacancies, opaque matching scores, and generated resume statements that are not traceable to actual experience. The problem addressed in this work is to develop a system that extracts structured evidence from resumes, compares that evidence with job requirements using explainable scoring, identifies skill gaps, generates ordered learning guidance, verifies job-posting links, and ensures that AI-assisted resume tailoring remains grounded in the candidate's source information.

## III. EXISTING SYSTEM

Conventional job portals and resume tools typically provide vacancy search, keyword-based screening, resume templates, and automated suggestions. However, listings collected from multiple sources may be stale, duplicated, or redirected to aggregator pages. Keyword-based matching may not adequately recognize related skills expressed through synonyms, while a single match percentage may not explain why a candidate is suitable or unsuitable. General-purpose language-model resume writers can generate fluent content but may introduce unsupported technologies, metrics, responsibilities, or seniority. Many tools also provide lists of missing skills without arranging them into a prerequisite-aware learning plan. These limitations can lead to wasted applications, unclear decisions, and inaccurate resume claims.

## IV. PROPOSED SYSTEM

RoleRadar is proposed as an integrated resume intelligence platform that connects resume parsing, evidence-based job matching, skill-gap identification, learning roadmaps, and safe resume tailoring. The system uses a three-tier architecture with a React and TypeScript frontend, a FastAPI backend, and MongoDB persistence. Resume content is converted into provenance-tagged evidence units, job requirements are mapped to those units, and transparent rules calculate match and ATS-oriented scores. A language model is used only to rephrase supported content; a deterministic Truth Guard checks every rewrite and reverts unsupported output to the original evidence. The platform further provides role intelligence, prerequisite-ordered learning plans, verified employer-feed job discovery, and a user-scoped career assistant.

## V.RELATED WORK

### A. Person–Job Fit

Person–job fit asks how well a candidate's qualifications match a posting. Qin et al. learn word-level semantic representations of job requirements and candidate experiences from historical application data, with the aims of reducing manual effort and giving interpretable fit results [1]. Such models need large volumes of labeled hiring outcomes, which a student project does not have. RoleRadar therefore uses transparent rules and similarity measures whose every contribution can be shown to the user, trading predictive flexibility for inspectability. [10], [11], [12]

### B. Transparency in Algorithmic Hiring

Raghavan et al. examined what vendors of algorithmic pre-employment assessments disclose about development, validation, and bias mitigation, and discussed the technical and legal questions that follow [3]. RoleRadar's design-level response is to publish its weights, display every factor, and treat undisclosed salary or experience as neutral rather than negative. It remains a decision-support tool, and Section VI discusses the fairness questions it leaves open. [17]

### C. Hallucination and Grounded Generation

Ji et al. survey hallucination in natural language generation, including source-conditioned tasks where output should remain faithful to its input [2]. Retrieval-augmented generation pairs a language model with an external non- parametric memory to produce more factual output [4]. Conditioning on source text does not by itself guarantee faithfulness, so RoleRadar adds a post-generation check: each rewritten bullet is compared with the claim it came from, and a failing rewrite is replaced by the source. [13], [14]

### D. Layout-Aware Resume Parsing

Classical document layout analysis segments pages with projection profiles; the recursive X–Y cut of Ha et al. applies this idea to the bounding boxes of connected components [5]. RoleRadar reads native PDFs, where text blocks and coordinates are already available, so it needs only one decision per page: do two genuine columns exist? A five-gate geometric test answers this question without document-specific rules (Section IV-A).

## VI.SYSTEM ARCHITECTURE AND IMPLEMENTATION

Role Radar is a three-tier web application (Fig. 1). The client is a single-page application written in React 19 and TypeScript, built with Vite and styled with Tailwind. Its routes cover onboarding, a dashboard, opportunity discovery, resume tailoring, a growth view for skills and roadmaps, a copilot chat, and settings. The client calls an asynchronous FastAPI service over REST and JSON with a bearer token, and the service persists data in MongoDB through the Motor driver.

The backend is organized into independent modules: authentication, profile, resume, jobs, matching, intelligence, tailoring, learning, interview, copilot, applications, and dashboard. Two cross-cutting services support them. The AI service layer hides the model behind a provider interface with local (Ollama, LM Studio), cloud-fallback, and mock implementations. Structured calls are validated against a schema and retried with a repair instruction up to two times before failing. An

embedding service provides sentence embeddings [6] or, when configured, TF-IDF vectors [7]. The copilot chat answers from a context assembled server-side from the authenticated user's own profile, evidence, matches, and applications, so that no other user's records enter the prompt.

Security follows common practice: bcrypt password hashing, HS256-signed JSON Web Tokens with a 24-hour lifetime, per-route rate limiting, a 5 MB upload limit, and queries scoped to the authenticated user.

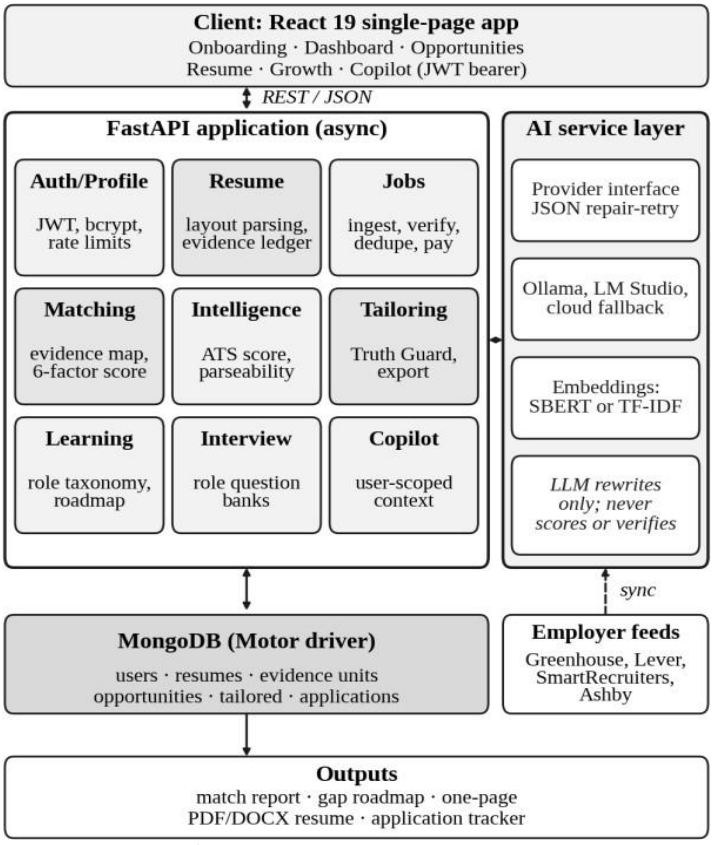



*Fig. 1. System architecture of RoleRadar. Shaded modules carry the core evidence pipeline.*

The backend uses FastAPI 0.115, Pydantic 2.9, Motor 3.6, PyMuPDF 1.24, python-docx, ReportLab, and spaCy 3.7 with phrase matching for skill extraction. The frontend uses React 19, Vite 8, Tailwind 4, and TypeScript 6. PDF exports are rendered with ReportLab and DOCX exports with python-docx in five layouts: standard, classic, Harvard, executive, and compact. The repository contains about 33,000 lines of backend Python, 33,000 lines of Python tests, and 15,000 lines of TypeScript.

## VII.METHODOLOGY

### A. Layout-Aware Parsing and the Evidence Ledger

Resumes are read with PyMuPDF, which returns text blocks with bounding boxes. Blocks wider than 70% of the page are treated as full width. For the remaining blocks, two-column reading order is applied only if all five gates hold: (1) the left cluster's right edge and the right cluster's left edge are at least 50 page units apart; (2) the right column starts beyond 38% of the page width; (3) each column holds at least two blocks; (4) neither column is more than five times taller than the other; and (5) the columns overlap vertically by at least 15% of the shorter one. If any gate fails, the page is read as a single stream, sorted in 8-unit vertical bands and then left to right. The gates use only geometry, so no vocabulary or template is assumed.

The ordered text is segmented into sections and converted into a candidate profile whose basic unit is the *evidence unit*: one claim with an identifier, section, claim type (for example delivery, achievement, or qualification), the technologies and metrics it mentions, and the employer or project it belongs to. Each skill receives an evidence tier (experience, project, explicit listing, weak mention, inferred, or unsupported) according to where it is demonstrated. Candidates are classified as fresher, experienced, career switcher, or internship seeker, and this category selects the scoring weights in Section IV-C. A parseability score starts at 100 and subtracts fixed penalties of 5 to 30 points for layout features that ATS parsers handle poorly.

### B. Requirement-to-Evidence Mapping

A job description is parsed into requirements, each tagged must-have, preferred, or contextual and given an importance weight between 0.5 and 1.0. Each requirement is compared with the profile and assigned one of eight support levels (Table I). An exact match requires every detected skill plus at least one delivery unit, that is, an experience or project bullet that demonstrates it. A skill that is only listed, or that appears only in education, earns a lower level. Compound requirements with partial coverage score 0.40 + 0.45 m/n, where m of n skills are supported. Hard constraints the profile cannot meet, such as a clearance or citizenship requirement, yield a conflicting status. The overall evidence score is the importance-weighted mean

in (1), where $\omega_i$ is the weight and $e_i$ the level score of requirement i. [15], [16]

$$E = 100 \cdot \Sigma_i \, \omega_i \, e_i \, / \, \Sigma_i \, \omega_i \qquad (1)$$

| Level | Score | Typical condition |
|---|---|---|
| Exact match | 1.00 | All skills found, with delivery evidence |
| Strong match | 0.95 | Synonym or mixed match (e.g., K8s for Kubernetes) |
| Supported | 0.80–0.85 | Skill listed (0.85) or education only (0.80) |
| Related | 0.70 | Closely related technology present |
| Partial | 0.40–0.85 | Part of a compound requirement; half the required tenure (0.60); text overlap (0.50) |
| Weak | 0.30 | Single weak textual overlap |
| Missing | 0 | No supporting evidence |
| Conflicting | 0 | Hard constraint not met |

*TABLE I. EVIDENCE SUPPORT LEVELS*

### C. Match Scoring and Readiness

Six factor scores $s_k$ in [0, 100] are combined as in (2), with weights $w_k$ from Table II.

$$S = \Sigma_k \, w_k \cdot s_k \qquad (2)$$

The skill score gives full credit to an exact match and 0.6 credit to a required skill whose best embedding similarity to any candidate skill is at least 0.55, as in (3), where M, P, and R are the matched, partially matched, and required skill sets.

$$s_{\text{skill}} = 100 \cdot (|M| + 0.6|P|) \, / \, |R| \qquad (3)$$

The role score is the best similarity between the candidate's target roles and the job title (50 if none is stated). The experience score is 100 inside the posted range, falls by 25 points per missing year, and falls by 10 points per surplus year down to a floor of 40. Location, compensation, and industry follow simple rules, for example 100 for a remote posting when the candidate prefers remote work. When a posting does not disclose pay or an experience range, the factor is set to a neutral 50, so that missing information does not hide an otherwise good opportunity. TABLE II. MATCH WEIGHTS BY CANDIDATE CATEGORY

| Category | Skill | Role | Exp. | Loc. | Pay | Ind. |
|---|---|---|---|---|---|---|
| Fresher | 0.40 | 0.20 | 0.05 | 0.15 | 0.15 | 0.05 |
| Experienced | 0.30 | 0.20 | 0.25 | 0.10 | 0.10 | 0.05 |
| Career switcher | 0.35 | 0.15 | 0.10 | 0.15 | 0.15 | 0.10 |
| Internship | 0.45 | 0.15 | 0.05 | 0.20 | 0.15 | 0.00 |
| Default | 0.35 | 0.20 | 0.15 | 0.15 | 0.10 | 0.05 |

*TABLE II. MATCH WEIGHTS BY CANDIDATE CATEGORY*

Freshers and internship seekers weight skills most heavily and experience least, whereas experienced candidates weight experience at 0.25. The final score maps to a verdict: ready (90 or above), fix gaps (70 to 89), or learn first (below 70). Every response carries the factor weights and a plain-language explanation, so the number can be reproduced by hand.

### D. ATS-Oriented Scoring

A separate 100-point ATS score estimates how a screening system might read the resume against a job description (Table III). A knockout gate returns zero when the posting demands a citizenship or clearance that the resume does not mention. Keyword stuffing is discouraged: if a single keyword exceeds 3.2% of the resume's words, five points are deducted, and a stuffing warning is raised above 3.5%. The score is a heuristic estimate, not a simulation of any vendor's parser.

| Component | Points | Basis |
|---|---|---|
| Technical keywords | 40 | Exact match 25, semantic coverage 15 |
| Experience context | 30 | Role alignment 15, action verbs and metrics 15 |
| Education and domain | 15 | Education level 10, domain knowledge 5 |
| Parsing integrity | 15 | Section headers 10, date formatting 5 |

*TABLE III. ATS SCORE COMPONENTS*

### E. Truth-Guarded Tailoring

Tailoring (Fig. 2) begins with a plan. The candidate's career stage, one of 17 labels, maps to one of eight strategies. The strategy fixes a template family (four ATS-friendly families), a content density (low, medium, or high), standard ATS headings, and per-section bullet budgets. The LLM then proposes rewrites of individual bullets for the target posting.

Every proposal is passed to deterministic checks that compare it with its source: (a) fabricated technical terms that

appear in the proposal but nowhere in the source profile; (b) unsupported metrics, and metrics that cross from one project or employer to another; (c) dropped source skills; (d) escalated seniority or title terms; (e) leadership, deployment, optimization, or architecture claims, and production-scope modifiers, without source support; and (f) sentence fragments, orphaned continuations, lost leading verbs, and abrupt truncation. Protected fields such as contact details and location must be preserved. A ledger conservation check requires every source evidence unit to appear in the tailored profile or to be removed with an explicit reason; units that vanish silently are flagged as accidentally lost, and units with no source provenance are flagged as phantom.

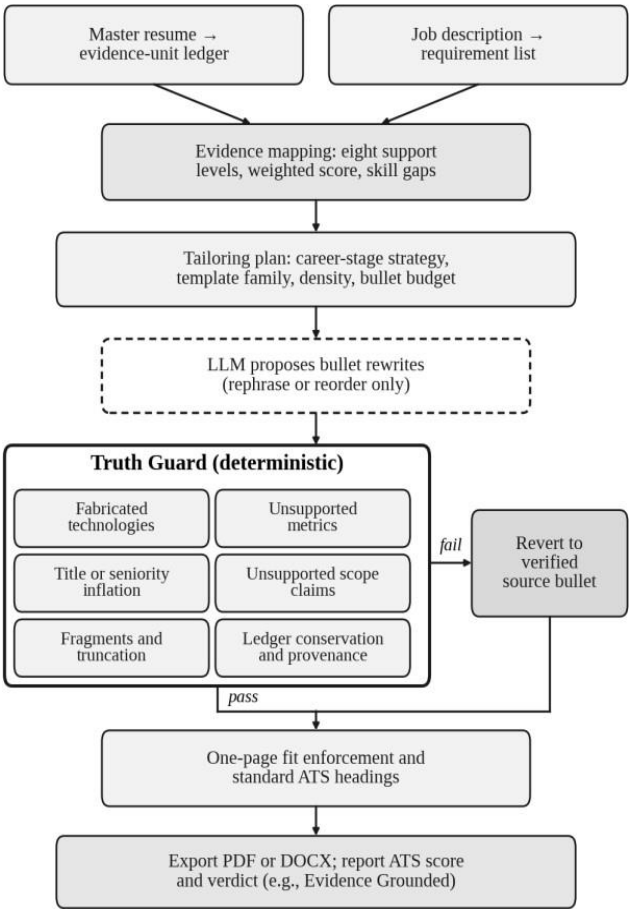



*Fig. 2. Truth-guarded tailoring pipeline. The LLM step is the only non-deterministic stage.*

By default, a unit that fails is reverted to its verified source claim, so the system fails safe: the worst outcome is an untailored bullet, never an invented one. A final step renders the document and enforces one-page fit by deterministic, priority-ordered trimming that never removes bullets containing metrics or required skills. The result carries a verdict (for example, Evidence Grounded) and is exported as PDF or DOCX in one of five layouts.

### F. Role Intelligence and Roadmaps

The role taxonomy holds 119 canonical roles in 25 domains, plus 28 controlled specializations. Each profile lists aliases, competencies by importance, tools, and knowledge areas. The resolver normalizes the input, expands compound titles, and checks canonical names, aliases, and approved modifiers. Generic words such as engineer, analyst, or manager can never decide a role alone, and an unknown modifier (for example, marine on a robotics title) returns low confidence with no skills instead of a software default. Missing competencies become skill gaps ranked core, secondary, or bonus, with planning estimates of 10, 5, and 3 days respectively.

Roadmap ordering uses a declarative prerequisite graph. Skills are scheduled with Kahn's algorithm [8]: at each step the ready skill with the highest priority is chosen, and a prerequisite that unlocks a core skill is promoted so that foundations precede advanced topics. If a cycle exists, the remaining skills are appended in priority order. This replaced an earlier chunking scheme that had placed predictive modeling before Python and SQL for data science roles (Section V-B).

### G. Verified Opportunity Pipeline

Postings are pulled from public employer job-board feeds (Greenhouse, Lever, SmartRecruiters, and Ashby) and normalized; the Greenhouse Job Board API, for example, exposes published postings without credentials [9]. A URL classifier labels each application link as a direct requisition, aggregator redirect, corporate portal, search-results page, invalid, or unverified. A posting is verified-active only when it has a direct requisition URL and is no more than 45 days old. Verified postings that have not been re-checked for 14 days become stale, and those past 45 days expire. Further states are pending,

closed, invalid, and market benchmark; the last holds 55 curated reference postings that are kept out of live results. Postings are rejected for reasons such as a non-Indian location, an insufficient description, a missing apply link, duplication, or an untrusted source.

Compensation is extracted from structured feed fields first and from description text second; patterns screen out company revenue, loan amounts, and years of experience, and when nothing is disclosed the value stays empty and is labeled undisclosed. An eligibility engine compares degree, graduation year, experience, and location with each posting and returns eligible, likely eligible, or a specific mismatch, so that a staff-level role is not presented as a good fit to a student.

## VIII.EVALUATION

The project has no labeled hiring outcomes, so evaluation consists of automated testing, internal validation harnesses, and an end-to-end acceptance run. These show that the system behaves as specified, not that it improves interview outcomes.

### A. Results

The reported evaluation produced the following outcomes: 1,235 of 1,268 backend regression cases passed (97.4%), with 29 failures and 4 skipped cases. Internal role-resolution validation resolved all 158 realistic job titles with high or medium confidence. The end-to-end acceptance run passed 17 of 18 checks. In the Data Engineer case study, the overall match score was 31, the skill score was 0, and two skill gaps—Microsoft Azure and Splunk—were identified. The tailored resume received an "Evidence Grounded" verdict, with zero fabricated claims detected by the implemented checks, an ATS parseability score of 100, and a recruiter-impact score of 76. These results indicate implementation and workflow consistency; they do not establish hiring effectiveness or real-world resume-matching accuracy.

### B. Regression Tests

The backend suite has 118 test modules containing 1,071 test functions, which pytest expands to 1,268 cases. A clean re-execution on Linux with the mock AI provider and TF-IDF embeddings passed 1,235 cases (97.4%), failed 29, and skipped 4. The failures we inspected involve a Windows-style path in an upload-sanitization test, an assertion about the default embedding provider that depends on local configuration, and curated-seed job tests. All 29 are under triage. Because the mock provider was used, the suite does not exercise a real LLM.

### C. Internal Validation

Table IV summarizes the validation harnesses kept in the repository. They were written by the project team, and the quality rubrics they apply, such as an average role-quality score of 99.3 out of 100, are self-scored. We therefore treat them as regression guards rather than independent evidence of quality. The role-resolution result is the clearest improvement: coverage rose from 97 of 158 realistic titles to all 158 after controlled specializations were added.

| Check | Result |
|---|---|
| Realistic role titles resolved with high or medium confidence | 158 / 158 |
| Niche titles returning low confidence and no invented skills | 20 / 20 |
| Generic single words blocked from deciding a role | 7 / 7 |
| Contrasting role pairs kept distinct | 15 / 15 |
| Domain pairs with skill-set Jaccard overlap below 0.15 | 28 / 28 |
| Roadmap prerequisite inversions across 30 roles | 0 |
| End-to-end acceptance checks passed | 17 / 18 |

*TABLE IV. INTERNAL VALIDATION RESULTS (SELF-ASSESSED)*

### D. End-to-End Acceptance and Case Study

An acceptance script drives the live API with a student profile through 18 checks covering discovery, eligibility, score transparency, gap derivation, roadmaps, tailoring, ATS analysis, export, application tracking, lifecycle handling, user isolation, empty states, and contract consistency. Seventeen passed; browser-based visual validation was blocked by the environment. In this run a senior staff-engineer posting was rejected with an experience mismatch, while a Data Engineer posting hosted on SmartRecruiters was kept as likely eligible.

For that posting, using the default weights, the overall match was 31, with a skills score of 0 and a neutral experience score of 50. The system listed exactly two gaps derived from the description, Microsoft Azure and Splunk, and the roadmap placed Azure as immediate and Splunk in week one. The tailored resume received the verdict Evidence Grounded with zero fabricated claims, an ATS parseability score of 100, and a recruiter-impact score of 76, and both PDF and DOCX exports were produced, with per-user data isolation confirmed.

## E. System Screenshots

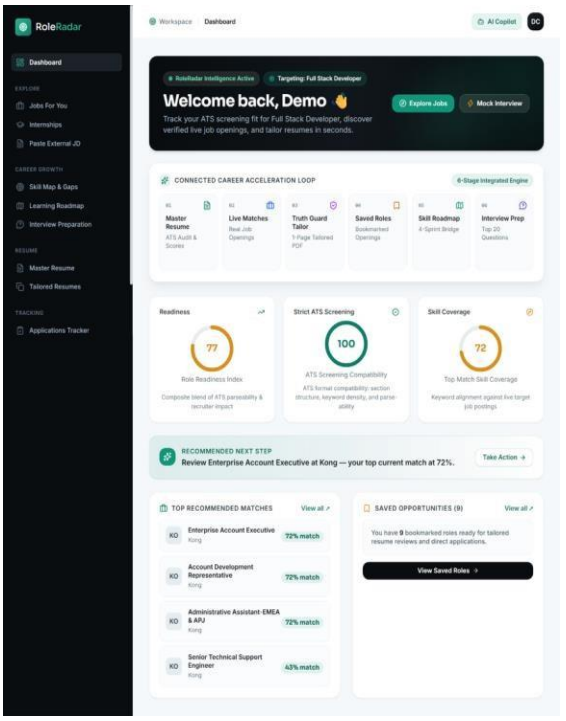



*Fig. 3. Fig. 1. RoleRadar dashboard summarizing readiness, ATS screening, skill coverage, recommended matches, and saved opportunities.*

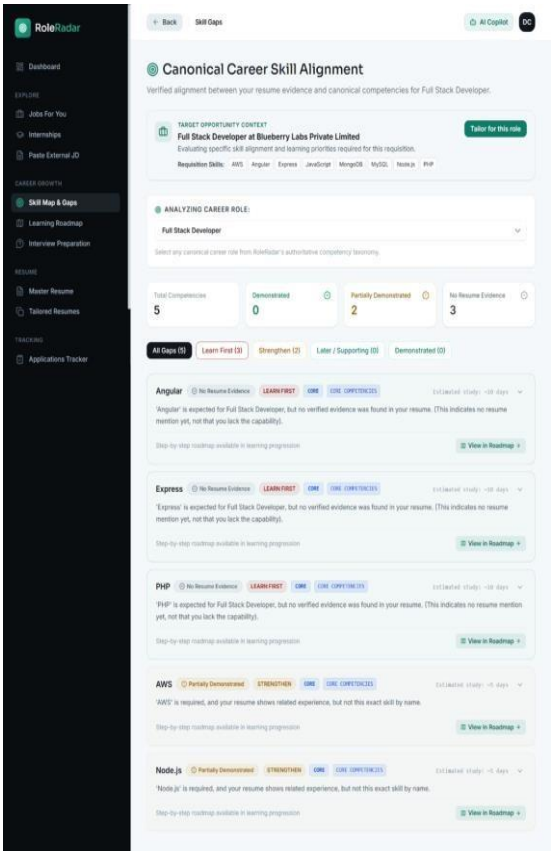



*Fig. 4. Fig. 2. Skill gap analysis interface showing demonstrated and missing competencies with learning-priority guidance*

.

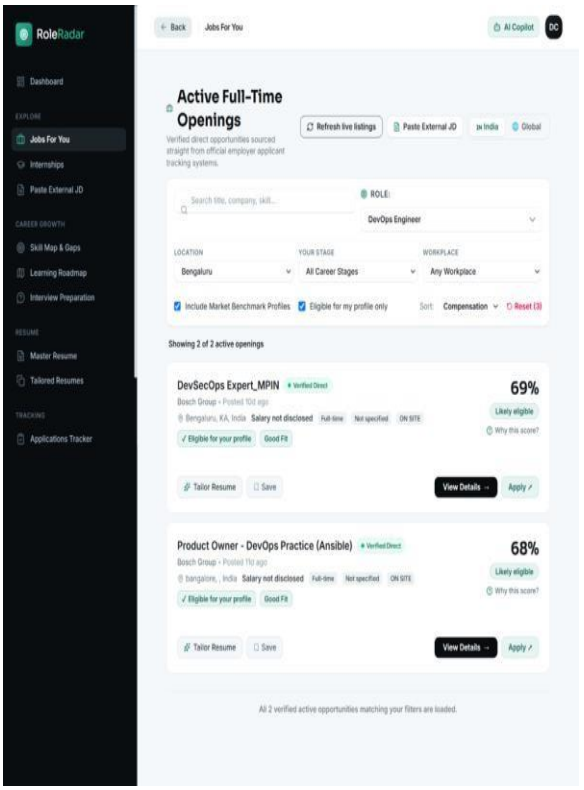



*Fig. 5. Fig. 3. Job recommendations interface displaying verified openings, eligibility indicators, match percentages, and application actions.*

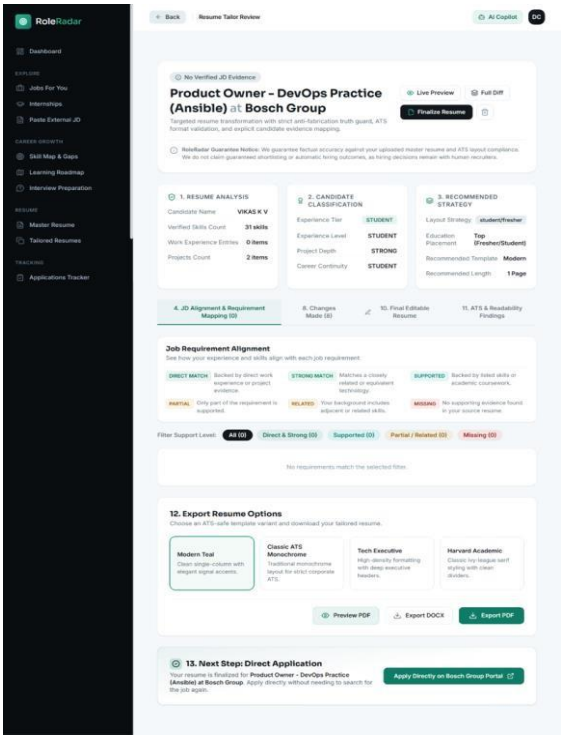



*Fig. 6. Fig. 4. Resume tailoring review interface showing candidate analysis, job requirement alignment, ATS findings, and resume export options.*

## IX.DISCUSSION AND LIMITATIONS

Separating deterministic verification from generation gave three practical benefits: scores can be reproduced by hand, failures are localized to a named rule, and a model failure degrades to an untailored bullet instead of a false one. The approach also has clear limits. [17]

- *No outcome evaluation.* There is no recruiter study or user study. Weights and thresholds, such as the 0.55 similarity cutoff, are hand-set and not calibrated against hiring outcomes.

- *Lexical Truth Guard.* The checks are rule-based. They can miss a fabrication that is phrased without a new technical term, and they can flag legitimate rewording when a synonym is absent from their tables. Precision and recall have not been measured on a labeled set.
- *Model dependence.* Rewrite quality depends on the configured LLM. The repository's own model-evaluation harness ran against the mock provider, so this paper makes no model-quality claim.
- *Coverage.* The job inventory is limited to configured employer boards and to Indian locations, and verification relies on public endpoints that can change.
- *Fairness.* Location and compensation factors can disadvantage some candidates. The ATS education component also awards more points for a computing degree (10 versus 7). Transparency lets users see why, but no disparate-impact audit has been done.

Automated application submission is not implemented; the platform prepares and tracks applications and leaves submission to the user.

## X.CONCLUSION

RoleRadar shows that a resume assistant can use an LLM without trusting it. Facts live in a provenance-tagged ledger, scores come from transparent rules, and every rewrite is verified against its source before export. Internal checks and an end-to-end run support the claim that the pipeline is consistent and auditable. Future work includes a labeled benchmark for Truth Guard precision and recall, a user study with students and recruiters, calibration of the match weights against outcomes, a fairness audit, and application submission with human review.

## REFERENCES


[1] C. Qin, H. Zhu, T. Xu, C. Zhu, C. Ma, E. Chen, and H. Xiong, "An enhanced neural network approach to person-job fit in talent recruitment," *ACM Trans. Inf. Syst.*, vol. 38, no. 2, pp. 15:1–15:33, 2020.

[2] Z. Ji et al., "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, pp. 248:1–248:38, 2023.

[3] M. Raghavan, S. Barocas, J. Kleinberg, and K. Levy, "Mitigating bias in algorithmic hiring: Evaluating claims and practices," in *Proc. ACM Conf. Fairness, Accountability, and Transparency (FAT*)*, 2020.

[4] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[5] J. Ha, R. M. Haralick, and I. T. Phillips, "Recursive X-Y cut using bounding boxes of connected components," in *Proc. 3rd Int. Conf. Document Analysis and Recognition (ICDAR)*, Montreal, Canada, Aug. 1995, pp. 952–955.

[6] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empirical Methods in Natural Language Processing and 9th Int. Joint Conf. Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, Nov. 2019, pp. 3982–3992.

[7] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.

[8] A. B. Kahn, "Topological sorting of large networks," *Commun. ACM*, vol. 5, no. 11, pp. 558–562, Nov. 1962.

[9] Greenhouse Software, "Job Board API." [Online]. Available: https://developers.greenhouse.io/job-board.html

[10] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.

[12] M. Zhang, K. N. Jensen, S. D. Sonniks, and B. Plank, "SkillSpan: Hard and soft skill extraction from English job postings," in Proc. 2022 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, WA, USA, 2022,pp. 4962–4984, doi: 10.18653/v1/2022.naacl-main.366.

[13] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? Explaining the predictions of any classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[15] T. Green, D. Maynard, and C. Lin, "Development of a benchmark corpus to support entity recognition in job descriptions," in Proc. 13th Language Resources and Evaluation Conference, Marseille, France, 2022.

[16] A.-S. Gnehm, E. Bühlmann, H. Buchs, and S. Clematide, "Fine-grained extraction and classification of skill requirements in German-speaking job ads," in Proc. Fifth Workshop on Natural Language Processing and Computational Social Science, Abu Dhabi, UAE, 2022,pp. 14–24

[17] E. Tabassi, "Artificial intelligence risk management framework (AI RMF 1.0)," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST AI 100-1, 2023, doi: 10.6028/NIST.AI.100-1.