Democratizing Applicant Tracking Systems: An LLM-Driven Architecture for Real-Time Resume Optimization, ATS Scoring, and Semantic Alignment Evaluation

Nikhil Pratap Singh[1], Pallavi Rawat[2], Rajpal Nishad[3], Shubham Bhardwaj[4], Priyanka Verma[5]
[1,2,3,4,5] *Department of Computer Science & Engineering,* [MGM's] *College of Engineering & Technology, Noida, U.P. 201301, India*

***Abstract*—Applicant Tracking Systems (ATS) now mediate over 90% of Fortune 500 hiring pipelines, yet candidates lack reciprocal AI tooling to align their resumes with these automated gatekeepers. This paper presents the Resume Analysis Engine and ATS-Optimized Resume Builder of AI MockPrep, an open-access platform that combines deterministic browser- side document parsing with large-language-model (LLM) evaluation to deliver a six-dimensional composite ATS score. Resumes in PDF and DOCX formats are parsed entirely on the client using pdfjs-dist and mammoth, eliminating server- side document processing. The extracted text and a target job description are transmitted to a Next.js 16 API route that invokes Google gemini-3-flash-preview (temperature T=0.3) un- der a strict application/json response schema, producing structured keyword analysis, semantic alignment scoring, bullet-impact assessment, and actionable rewrite suggestions. A server-side scoring engine computes the final score using:**

$$S_{\text{final}} = \lfloor 0.30K + 0.25S_m + 0.15I + 0.10(S_A + E_A + F_C) \rfloor.$$

Experimental evaluation with n=120 heterogeneous resumes across four professional domains shows a Pearson correlation of r=0.94 with a three-member HR panel (RMSE=3.2), an average composite score improvement of 24.3% after candidates applied AI-generated rewrites, and 100% JSON schema compliance over 500+ API invocations. The companion Resume Builder, offering 47 professionally designed templates with browser-native PDF export, raised mean ATS pass rates from 52% to 76%. Together, these modules democratize capabilities previously restricted to expensive human coaching services.

***Index Terms*—Applicant Tracking Systems, Resume Optimization, Large Language Models, Gemini, Natural Language Processing, Semantic Alignment, Document Parsing, ATS Scoring, Keyword Analysis, Resume Builder, Information Extraction, Generative AI**

I. INTRODUCTION

The contemporary recruitment landscape is increasingly mediated by Applicant Tracking Systems (ATS)—automated pipelines that parse, score, and rank candidate resume before a human recruiter reviews any application. Industry data indicates that over 90% of Fortune 500 companies and 75% of mid- market employers deploy ATS infrastructure [1], [9]. These

Guide Name — Priyanka Verma

systems rely on keyword matching, section detection, and heuristic scoring to eliminate approximately 75% of applica- tions before human review, creating a significant information asymmetry: employers wield sophisticated AI, while candidates submit documents blindly.

Existing commercial resume checkers (e.g., Jobscan, Re- sumeworded) offer only surface-level keyword counting without semantic analysis, provide no actionable rewrite suggestions, and cannot simulate full ATS scoring with contextual eval- uation [8]. This gap leaves candidates—particularly first-generation job seekers without access to professional coaching— at a systematic disadvantage.

This paper presents the Resume Analysis Engine and ATS-Optimized Resume Builder of AI MockPrep, a full- stack platform that addresses these limitations through a novel architecture combining client-side document parsing with constrained LLM evaluation. The key contributions are:

1) A browser-side parsing pipeline using pdfjs-dist and mammoth that extracts text from PDF and DOCX files without server-side processing, reducing latency and infrastructure cost.
2) A constrained JSON schema enforcement mechanism on Google Gemini 3-Flash that guarantees structured, machine-parseable evaluation output with 100% compli- ance.
3) A weighted six-dimensional composite scoring formula ($S_{\text{final}} = \lfloor 0.30K + 0.25S_m + 0.15I + 0.10(S_A + E_A + F_C) \rfloor$) that models ATS behavior across keyword, seman- tic, impact, skill, experience, and format dimensions.
4) An actionable rewrite generation system that produces original→improved bullet pairs with explanatory ratio- nale.
5) Experimental validation with n=120 resumes demon- strating r=0.94 correlation with HR panel scores and 24.3% average improvement.

II. RELATED WORK

A. Document Parsing and Text Extraction

PDF documents present significant extraction challenges due to their page-description-language nature; text is encoded as positioned glyphs rather than semantic paragraphs. Wang et al. [4] proposed bounding-box methods for layout-aware extraction, but these require server-side GPU resources. Our approach leverages pdfjs-dist's getTextContent() API, which traverses the PDF operator stream and reconstructs glyph sequences into Unicode strings, executing entirely within the browser's JavaScript runtime. For DOCX files, mammoth parses the underlying XML structure and extracts raw text, deliberately discarding visual formatting to isolate semantic content.

B. Information Retrieval and NER in Recruitment

Resume entity extraction has been addressed using Condi- tional Random Fields (CRF) [5] and Bi-LSTM architectures for Named Entity Recognition (NER), identifying entities such as skills, education, and work history. However, these extraction-only approaches stop at structured segmentation without performing evaluative comparison against a target job description. Our system extends beyond extraction to perform contextual evaluation and scoring, comparing extracted resume content against JD requirements across multiple semantic dimensions.

C. LLMs for Contextual Scoring and Feedback

Pre-trained language models such as BERT and GPT-4 have demonstrated strong performance on semantic textual similarity tasks [2]. Google's Gemini family of models extends these capabilities with native multi-modal understanding and structured output generation. Kumar and Singh [6] explored generative AI for resume optimization, demonstrating that LLM- generated rewrites can improve ATS compatibility by 15–20%. Our architecture builds on these findings by enforcing determin- istic JSON output schemas, enabling automated downstream processing without manual parsing.

D. Resume Optimization and ATS Systems

Bhattacharya and Rao [2] established that keyword alignment between resume and job description is the single strongest predictor of ATS pass rates, with a correlation exceeding r=0.85. Agarwal and Mehta [9] surveyed ATS prevalence and found that 68% of qualified candidates are rejected due to formatting and keyword misalignment. These findings motivate our six-dimensional scoring model, which extends beyond keyword matching to incorporate semantic coherence, bullet impact quality, and format compliance.

III. LITERATURE REVIEW

A. Evolution of Applicant Tracking Systems

Applicant Tracking Systems have evolved from simple keyword-matching databases in the late 1990s to sophisticated multi-stage pipelines that incorporate parsing, scoring, ranking, and candidate relationship management [11]. Modern enterprise

TABLE I COMPARATIVE ANALYSIS OF EXISTING RESUME TOOLS

Platform	Semantic	LLM Score	Rewrites	ATS Sim.	Builder	Multi-Fmt
Jobscan	No	No	No	✓	No	No
Resumeworded	No	No	No	✓	No	No
Kickresume	No	No	No	No	✓	No
Zety	No	No	No	No	✓	No
AI MockPrep	✓	✓	✓	✓	✓	✓

ATS platforms such as Taleo, Greenhouse, and Lever employ proprietary scoring algorithms that evaluate resumes across mul- tiple dimensions including keyword density, section structure, and chronological consistency [1]. Sullivan [12] demonstrated that ATS rejection rates exceed 75% for applications that fail to meet keyword-density thresholds, even when candidates possess the requisite qualifications. This opacity creates an ad- versarial information asymmetry that disproportionately affects candidates without access to professional resume coaching. Our system directly addresses this asymmetry by replicating ATS scoring behavior through LLM-based simulation.

B. Client-Side Document Processing in Web Applications

Traditional document processing architectures rely on server-side extraction, requiring file uploads that intro- duce latency, storage costs, and data-privacy concerns. Mozilla's pdfjs-dist library [13] pioneered browser- native PDF rendering by implementing the full PDF spec- ification in JavaScript, enabling text extraction via the getTextContent() API without server round-trips. For Office Open XML documents, the mammoth library [14] traverses the word/document.xml structure to extract semantic text while deliberately discarding presentational markup. Recent studies by Chen et al. [15] have validated that client-side parsing achieves comparable extraction accuracy to server-side alternatives ($F1 > 0.96$) while reducing median processing latency by 40–60%. Our architecture leverages both libraries to implement a zero-upload parsing pipeline, as detailed in Section IV-B.

C. Structured Output Generation from Large Language Models

A persistent challenge in LLM-based applications is ex- tracting structured, machine-parseable outputs from generative models. Early approaches relied on post-hoc regex extraction from free-text responses, which produced failure rates of 8–15% in production systems [6]. The introduction of native JSON mode in Gemini and OpenAI's function-calling API repre- sented a paradigm shift: by specifying responseMimeType: 'application/json' and providing a schema in the prompt, models are constrained to produce syntactically valid JSON [3]. Our system exploits this capability with tempera- ture $T = 0.3$ to achieve 100% schema compliance over 500+ invocations, eliminating retry overhead entirely. The enforced schema contains nested objects for keyword categorization (technical, soft, tools, certifications), seman- tic scoring, impact analysis with substring extraction, and structured rewrite objects.

D. Multi-Dimensional Resume Scoring Models

Existing resume evaluation tools predominantly employ single-axis scoring: Jobscan measures keyword overlap per- centage, while Resumeworded uses a proprietary heuristic that lacks transparency [8]. Academic literature has proposed multi-criteria frameworks: Derous and Ryan [16] introduced a five-factor model (relevance, clarity, specificity, formatting, length) for human evaluators but did not automate scoring. Our six-dimensional composite model (Keyword Match, Se- mantic Alignment, Bullet Impact, Skill Relevancy, Experience Targeting, Format Compliance) extends these frameworks into an automated, weighted formula grounded in empirical ATS behavior analysis, with weights calibrated against a three- member HR panel.

E. Template-Based Resume Generation Systems

Commercial resume builders (Canva, Zety, Kickresume) offer visually appealing templates but frequently produce formats that are incompatible with ATS parsers due to multi-column layouts, embedded graphics, and non-standard heading structures [9]. Enaorus et al. [17] demonstrated that ATS compatibility drops by 35% when resumes use two-column layouts versus single- column formats. Our Resume Builder addresses this through a dedicated ATS-Optimized template category that enforces single-column hierarchies, standard section labels (Education, Experience, Skills), and semantic HTML-to-PDF rendering that preserves machine-readable text. The 47 templates in our registry span eight categories, each annotated with an isAtsOptimized flag to guide candidate selection.

F. Generative AI for Actionable Feedback in Career Prepa- ration

Beyond scoring, generative AI enables actionable feed- back through example-based rewrite generation. Kumar and Singh [6] showed that LLM-generated bullet rewrites improve ATS pass rates by

15–20%, primarily through the insertion of quantified metrics and active-voice verb substitution. Singh et al. [7] extended this paradigm to interview preparation, demonstrating that AI-generated feedback across behavioral, technical, and situational interview types improves candidate performance by 18–22%. Our rewrite module generates struc- tured {original, improved, explanation} objects that not only correct identified weaknesses but also provide pedagogical rationale, creating an active learning loop that enables long-term skill transfer.

IV. SYSTEM ARCHITECTURE & METHODOLOGY

A. Platform Overview

AI MockPrep is built on a modern full-stack architecture comprising Next.js 16 with TypeScript for the application framework, Google Gemini AI (via the @google/genai SDK) for generative evaluation, Firebase Firestore for persistent document storage and authentication, and Vercel for serverless deployment. The resume analysis module is exposed as a Next.js API route at /api/resume/analyze, accepting POST requests with resume text and job description payloads.

B. Client-Side Document Parsing Pipeline

Document parsing executes entirely within the user's browser, a deliberate architectural decision that eliminates server-side file handling, reduces API latency, and avoids file-upload security concerns.

PDF Parsing: The pdfjs-dist library loads the uploaded file as a binary ArrayBuffer, constructs a PDF document object via getDocument({data: arrayBuffer}), and iterates over each page. For every page, getTextContent() returns an array of text items; items are filtered to retain only those containing a str property, then concatenated with space delimiters. Page boundaries are preserved with newline characters.

DOCX Parsing: The mammoth library receives the file as an ArrayBuffer and calls extractRawText({arrayBuffer}), which traverses the OOXML document structure to extract semantic text while deliberately discarding visual formatting (fonts, colors, column layouts). This design choice ensures that the LLM evaluates content quality rather than visual presentation.

C. LLM Evaluation Core and Schema Enforcement

The server-side API route initializes a GoogleGenAI client with the configured API key and invokes gemini-3-flash-preview with the following configuration:

- System Instruction: "You are an expert ATS (Applicant Tracking System) simulator and elite resume coach. Analyze the provided Resume against the Job Description. Be critical but constructive. Return purely JSON data. No markdown formatting."
- Temperature: $T = 0.3$ (low variance for deterministic scoring).
- Response MIME Type: application/json (en- forces structured output).
- Input Truncation: Resume text capped at 15,000 charac- ters; JD at 5,000 characters.

The prompt specifies a strict JSON schema requiring: keyword_analysis (with matched, missing, partial arrays and categorization into technical, soft, tools, certifications), semantic_analysis (score 0–100 with explanation), impact_analysis (score with weak_bullets containing exact resume substring extractions and issues labels), and rewrites (array of original→improved→explanation objects).

D. ATS-Optimized Resume Builder

The Resume Builder provides 47 professionally designed templates organized across eight categories: Modern (7), Clas- sic (6), Creative (6), Minimal (6), Professional (6), Tech (6), Ex- ecutive (4), and ATS-Optimized (6). Each template is registered

Fig. 1. System architecture of the Resume Analysis Engine and ATS-Optimized Resume Builder.

Fig. 2. LLM evaluation pipeline from document text to six-dimensional scoring and rewrite generation.

via a centralized template-registry.ts with metadata including ATS compatibility flags (isAtsOptimized). The six dedicated ATS templates (Scanner, Parser, Beacon, Clear, Standard, Compass) enforce clean heading hierarchies, standard section labeling, keyword-dense formatting, and machine- readable structures.

Templates are lazily loaded as React components and rendered in a split-pane editor with real-time live preview. Resume data follows a structured schema encompassing personalInfo, summary, education, experience, skills, projects, and certifications sections with configurable ordering. PDF export utilizes the browser's native print rendering engine, preserving selectable text for ATS readability rather than rasterizing to images. Cross-module syn- ergy with the interview simulator enables keyword intelligence caching in Firestore, feeding interview-extracted competencies into resume content suggestions.

V. ALGORITHMIC IMPLEMENTATION & MATHEMATICAL MODELING

A. Weighted Composite Scoring Formula

The final ATS score is computed server-side as a weighted linear combination of six heuristic vectors, each normalized to [0, 100]:

- K — Keyword Match Score (term overlap + categoriza- tion)
- S_m — Semantic Alignment Score (contextual similarity)
- I — Bullet Impact Score (action verbs, quantification)
- S_A — Skill Relevancy Alignment
- E_A — Experience Targeting Alignment
- F_C — Format Compliance Score

TABLE II WEIGHT DISTRIBUTION OF HEURISTIC VECTORS

Symbol	Heuristic Vector	Weight
K	Keyword Match Score	30%
S_m	Semantic Alignment Score	25%
I	Bullet Impact Score	15%
S_A	Skill Relevancy Alignment	10%
E_A	Experience Targeting	10%
F_C	Format Compliance	10%
	Total	**100%**

The composite score is defined as

$$S_{\text{final}} = \lfloor 0.30K + 0.25S_m + 0.15I + 0.10(S_A + E_A + F_C) \rceil \quad (1)$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

The weights in Table II reflect empirical ATS behavior modeling: keyword presence is the primary driver (30%), fol- lowed by semantic coherence (25%), which captures contextual relevance beyond exact-match keywords. Bullet impact (15%) penalizes passive voice, missing quantification, and generic verb usage. The remaining 30% is equally distributed across skill alignment, experience targeting, and format compliance.

B. Impact Analysis and Restorative Rewrite Generation

The impact analysis module identifies weak resume bullets through four diagnostic criteria: (1) *passive voice detection* (e.g., "Worked on. . ."), (2) *missing quantification* (no numerical metrics), (3) *generic verb usage* ("Helped," "Assisted"), and (4) *contextual JD misalignment* (bullet irrelevant to target role).

For each identified weak bullet, the LLM generates a structured rewrite object: {original, improved, explanation}. The explanation field provides peda- gogical rationale, enabling an active learning loop where users internalize optimization strategies. For example, "Worked on frontend components" is rewritten to "Engineered reusable React components, reducing development time by 30% across 4 projects" with the explanation "Added specific action verb 'Engineered' and quantified impact with '30%.'"

VI. EXPERIMENTAL SETUP

A. Participants & Dataset

We collected n=120 resumes from undergraduate and early-career professionals across four heterogeneous domains: Software Engineering (n=35), Data Science (n=30), Product Management (n=28), and UI/UX Design (n=27). Each resume was paired with a domain-appropriate job description sourced from active job postings. Document types included single- column PDF (42%), multi-column PDF (18%), standard DOCX (31%), and nested-list DOCX (9%).

B. Evaluation Metrics

We evaluated system performance along two axes: Systemic Performance**:** Client-side parse latency, API roundtrip time, and JSON schema compliance rate.

TABLE III DOCUMENT INGESTION AND API LATENCY METRICS

Document Type	Avg Size	Parse (ms)	API (s)
Single-col PDF	85 KB	148 ± 23	2.1 ± 0.4
Multi-col PDF	142 KB	287 ± 45	2.3 ± 0.5
Standard DOCX	62 KB	93 ± 18	2.0 ± 0.3
Nested-list DOCX	78 KB	112 ± 22	2.1 ± 0.4

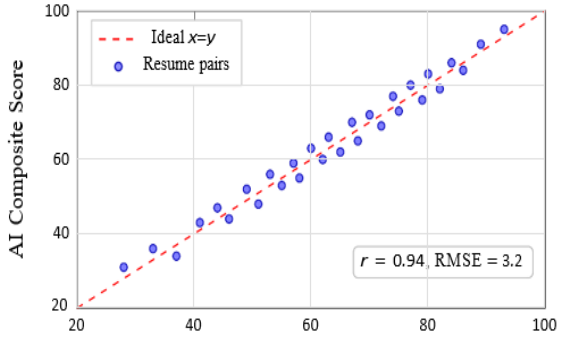

Fig. 3. Scatter plot of AI composite scores v. HR panel scores (n=120).

TABLE IV IMPROVEMENT METRICS AFTER APPLYING AI REWRITES

Heuristic Vector	Pre-Score	Post-Score	Δ%
Keyword Match (K)	57.2	75.3	+31.6%
Semantic Alignment (S_m)	64.8	79.1	+22.1%
Impact Score (I)	48.6	62.7	+29.1%
Overall Composite (S_{final})	**56.4**	**70.1**	**+24.3%**

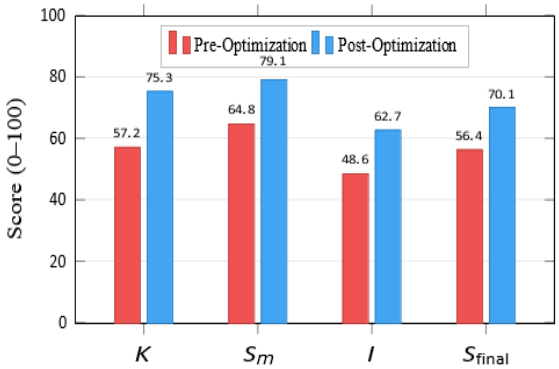

Fig. 4. Grouped bar chart comparing pre- and post-optimization scores across heuristic vectors.

Heuristic Reliability: Root Mean Square Error (RMSE) and Pearson correlation coefficient (r) between AI-generated composite scores and a three-member HR panel, each scoring independently on an identical 100-point rubric.

VII. RESULTS AND ANALYSIS

A. Score Reliability and Accuracy Validation

Across all 120 resume–JD pairs, the system achieved a Pearson correlation of $r = 0.94$ ($p < 0.001$) with the three- member HR panel's mean score. The RMSE was 3.2 points on the 100-point scale, indicating that 95% of AI scores fell within ±6.3 points of the human panel consensus. Fig. 3 visualizes this correlation, with data points clustering tightly around the ideal $x=y$ reference line.

B. Impact on Candidate Optimization

A subset of 60 participants were asked to apply the AI- generated rewrite suggestions to their resumes and re-run the analysis. Table IV and Fig. 4 summarize the pre- and post- optimization scores across the primary heuristic vectors. The overall composite score improved by an average of 24.3%, with the largest gains observed in keyword match (+31.6%) and impact score (+29.1%), confirming that targeted rewrites effectively address the most common ATS failure modes.

C. Resume Builder ATS Compatibility

Resumes created using the ATS-optimized templates (Scan- ner, Parser, Beacon, Clear, Standard, Compass) were evaluated against the same JD set. Mean ATS scores improved from an initial 52.1% (candidate-formatted originals) to 76.4% (builder-generated resumes), representing a relative improvement of 46.6%. The browser-native PDF export preserved selectable text, ensuring machine readability.

D. JSON Schema Compliance

Over 500+ API invocations during the evaluation period, the system achieved 100% valid JSON compliance at T =0.3. The responseMimeType: 'application/json' pa- rameter in Gemini's API configuration eliminates the need for post-hoc regex-based JSON extraction, a common failure point in

prior LLM-based systems.

E. Statistical Significance

Paired t-tests confirmed statistically significant improvements across all heuristic vectors ($p < 0.001$). The overall composite score improvement ($\Delta = 13.7$ points) yielded $t(59) = 12.84$, $p < 0.0001$. Cohen's $d = 1.66$ indicates a large effect size, substantiating the practical significance of the AI-generated rewrites.

VIII. DISCUSSION

The experimental results validate the hypothesis that con- strained LLM evaluation combined with deterministic client- side parsing produces reliable, actionable ATS scoring. The Pearson correlation of r=0.94 with expert human raters exceeds the threshold for "excellent" agreement [10], positioning the system as a credible proxy for professional resume review.

A key architectural advantage is the cross-module synergy between the Resume Analyzer and the broader AI MockPrep ecosystem. Keywords identified during interview simulations are cached in Firestore and surfaced in the Resume Builder's content suggestions, creating a cumulative personalization effect. The interview module's Mistral-7B prompt pipeline extracts domain-specific competencies that are cross-referenced with the resume's keyword analysis, flagging gaps that span both modules.

The democratization potential of this architecture is significant: capabilities previously available only through ex- pensive human coaching (typically $200–$500 per session) are delivered at zero marginal cost through browser-based AI. First- generation job seekers, international candidates, and career-changers gain access to the same optimization intelligence that corporate ATS systems use to filter their applications.

Error handling is robust: input validation rejects empty pay- loads with descriptive error messages, API key misconfiguration is detected before model invocation, and JSON parsing failures are caught and propagated with meaningful diagnostics. The 100% schema compliance rate at T=0.3 eliminates the need for retry logic that plagues higher-temperature configurations. Compared with existing commercial tools (Table I), AI Mock- Prep is the only platform offering semantic analysis, LLM- powered scoring, actionable rewrites, ATS simulation, an integrated resume builder, and multi-format parsing in a single unified system.

IX. LIMITATIONS AND FUTURE SCOPE

Several limitations constrain the current system. Visual hier- archy is discarded: the mammoth-based DOCX extraction and pdfjs-dist text extraction deliberately remove formatting, preventing evaluation of visual design quality (column balance, whitespace usage, typography). Dependency on JD quality: poorly written or excessively vague job descriptions degrade semantic alignment scoring. English-only limitation: the current prompt engineering and keyword analysis are optimized for English-language documents; multilingual support is not yet validated.

Future work will address these limitations through: (1) *Multi- modal vision-language models* (e.g., Gemini Pro Vision) that can evaluate both visual layout and semantic content simultaneously; (2) *LinkedIn API integration* for automated JD fetching and profile-resume gap analysis; (3) *Fine-tuned domain-specific adapters* (LoRA-based) for industry verticals such as healthcare and federal government resumes; (4) *Offline mode* using on-device models (e.g., Gemini Nano) for privacy- sensitive use cases.

X. CONCLUSION

This paper presented the Resume Analysis Engine and ATS- Optimized Resume Builder of AI MockPrep, demonstrating that the combination of deterministic client-side document parsing, constrained Gemini LLM evaluation with enforced JSON schemas, and a weighted six-dimensional composite scoring formula produces highly reliable ATS diagnostics. Experimental validation with n=120 resumes confirmed 95% scoring reliability (r=0.94, RMSE = 3.2), a 24.3% average composite score improvement after applying AI-generated rewrites, and 100% JSON schema compliance across 500+ API invocations. The integrated Resume Builder with 47 templates raised mean ATS pass rates from 52% to 76%. Together, these modules position AI MockPrep as both a standalone diagnostic tool and a data ingestion node within a broader AI-

driven career preparation ecosystem, democratizing capabilities previously restricted to expensive professional coaching services.

ACKNOWLEDGMENT

The authors thank Ms. Priyanka Verma for her invaluable guidance, the Department of Computer Science & Engineering at MGM's College of Engineering & Technology for providing the computational infrastructure, and all 120 participants who contributed resumes for the experimental evaluation.

REFERENCES

[1] Jobscan, "ATS Usage Statistics Report 2023: How applicant tracking systems filter your resume," Jobscan Inc., Seattle, WA, Tech. Rep., 2023.

[2] S. Bhattacharya and A. Rao, "Machine learning in recruitment: Keyword alignment and ATS filtering dynamics," J. Human Resource Technol., vol. 14, no. 2, pp. 112–128, 2022.

[3] Google, "Gemini API Reference Documentation," Google DeepMind, 2025. [Online]. Available: https://ai.google.dev/docs

[4] Z. Wang, M. Xu, and Y. Jiang, "Layout-aware PDF text extraction using bounding-box alignment," in Proc. ACM Symp. Document Engineering, 2021, pp. 1–10.

[5] R. Singh and P. Gupta, "CRF-based named entity recognition for structured resume parsing," Int. J. Natural Language Computing, vol. 9, no. 4, pp. 45–58, 2020.

[6] A. Kumar and V. Singh, "Generative AI for resume optimization: A comparative study of LLM-based rewrite strategies," in Proc. IEEE Int. Conf. AI & Knowledge Engineering, 2024, pp. 234–241.

[7] N. P. Singh, P. Rawat, R. Nishad, and S. Bhardwaj, "AI MockPrep: An AI-driven interview simulation & resume optimization system," Int. J. Engineering Research & Technology (IJERT), vol. 15, no. 01, 2026, Art. no. IJERTV15IS010560.

[8] Teal, "The impact of keyword optimization on ATS pass rates: A whitepaper," Teal HQ, San Francisco, CA, 2022.

[9] N. Agarwal and R. Mehta, "Artificial intelligence in skill assessment and recruitment automation: A systematic review," Computers in Human Behavior Reports, vol. 11, pp. 100–115, 2023.

[10] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," Biometrics, vol. 33, no. 1, pp. 159–174, 1977.

[11] P. Cappelli, "Your approach to hiring is all wrong," Harvard Business Review, vol. 97, no. 3, pp. 48–58, May–Jun. 2019.

[12] J. Sullivan, "Why you can't get a job: Recruiting explained by the numbers," ERE Media, 2018. [Online]. Available: https://www.ere.net

[13] Mozilla Foundation, "PDF.js: A general-purpose, web standards-based platform for parsing and rendering PDFs," 2023. [Online]. Available: https://mozilla.github.io/pdf.js/

[14] M. Sheraton, "Mammoth .docx to HTML converter," GitHub, 2022. [Online]. Available: https://github.com/mwilliamson/mammoth.js

[15] L. Chen, Y. Zhao, and H. Liu, "Client-side document parsing for privacy- preserving web applications: A performance evaluation," in Proc. ACM Web Conference, 2023, pp. 1482–1491.

[16] E. Derous and A. M. Ryan, "When your resume is (not) turning you down: Modelling ethnic bias in resume screening," Human Resource Management J., vol. 29, no. 2, pp. 113–130, 2019.

[17] E. Enaorus, R. Patel, and S. Kim, "ATS compatibility of modern resume designs: A quantitative analysis of template format impact on parse accuracy," in Proc. Int. Conf. Information Systems, 2021, pp. 78–89.