International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Skill Edge - Resume Analyzer with AI Interview Questions and Job Match

B Sujatha[1], Chowdari Nooka Naveenth[2], Kolli Yeshwanth Venkat Kumar[3], Davala Sudharsan Naidu[4], Killi Tarun[5]
[1]Department of Computer Science & Engineering – Data Science
Anil Neerukonda Institute of Technology and Sciences
Visakhapatnam, India

Abstract. The modern recruitment landscape demands tools that go far beyond simple keyword filtering. Organizations process thousands of job applications daily, yet the majority of automated screening systems in use today are incapable of understanding a candidate's true competency level or matching that competency against the genuine requirements of a specific role. This paper presents Skill Edge, a fully browser-deployed, AI-powered resume analysis platform that addresses three interconnected problems in modern hiring: resume quality evaluation, role-based job matching, and personalized interview preparation. The system accepts a PDF resume as input and processes it through a multi-stage pipeline comprising a document ingestion layer, a linguistic preprocessing engine, a competency recognition module, a weighted scoring engine, a job match computation unit, and a Groq-accelerated LLaMA 3.3 inference layer for generating candidate-specific interview questions. Skill Edge detects over one hundred technical skills, classifies candidates by career domain and experience tier, computes a one-hundred-point resume quality score across ten weighted dimensions, and produces a job compatibility percentage by comparing extracted competencies against role-specific skill profiles. The AI question generation module constructs a structured prompt from the candidate's detected profile and dispatches it to the Groq API, returning a personalized set of interview questions within seconds. Evaluation across a heterogeneous dataset of resumes from multiple domains confirms that Skill Edge outperforms conventional keyword-based systems in skill extraction accuracy, job match precision, and recommendation quality. The platform is publicly accessible at https://skill-edge-zeta.vercel.app/ with no login required.

Keywords: Artificial Intelligence, Natural Language Processing, Resume Analysis, Job Matching, Skill Gap Detection,Applicant Tracking System, LLaMA, Groq API

1 INTRODUCTION

The recruitment process is one of the most consequential yet operationally demanding functions within any organization. Identifying the right candidate from a pool of hundreds of applications requires careful evaluation of qualifications, experience, skills, and cultural fit, a task that is time-consuming, subjective, and difficult to scale. As organizations grow and hiring volumes increase, the pressure to automate candidate screening has driven widespread adoption of Applicant Tracking Systems (ATS). However, while these systems offer speed, they sacrifice depth. Most commercially deployed ATS platforms operate on rigid keyword matching, meaning a candidate's suitability is judged not by their actual abilities but by whether the words on their resume happen to match the words in a job description [1].

This limitation has well-documented consequences. Qualified candidates are routinely eliminated because they describe their experience using different terminology than what appears in a job posting. A data analyst who writes "built predictive models using Python" may be filtered out from a role requiring "machine learning experience" even though both phrases describe overlapping competencies. Simultaneously, unqualified candidates who have learned to insert the right keywords into their resumes can pass initial screening despite lacking genuine skills. The result is a system that neither accurately identifies strong candidates nor reliably excludes weak ones [2].

Beyond keyword matching, existing tools offer little to candidates themselves. A resume may be rejected, but the candidate receives no explanation of why, no understanding of which skills they are missing for their target role, and no guidance on how to improve. This information asymmetry creates a frustrating experience for job seekers and results in wasted effort on both sides of the hiring process.

Artificial Intelligence, and specifically Natural Language Processing (NLP), offers a fundamentally different approach. Rather than matching words, NLP-based systems can understand meaning, recognizing that "developed RESTful APIs" and "built web services" describe similar competencies, that "led a team of five engineers" signals project management experience, and that a resume mentioning TensorFlow, PyTorch, and Scikit-learn belongs to a machine learning domain profile [3]. When combined with

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

large language models capable of generating natural language, these systems can also produce personalized, context-aware feedback rather than generic responses.

This paper introduces Skill Edge, a production-deployed AI resume analyzer that brings together four distinct capabilities in a single browser-based application. First, it evaluates the quality of an uploaded PDF resume against a ten-dimension weighted rubric and produces a score out of one hundred. Second, it performs job matching by comparing the candidate's detected skill set against predefined role-specific competency profiles and computing a compatibility percentage. Third, it identifies skill gaps competencies that the target role requires but the candidate's resume does not demonstrate and presents them as prioritized learning recommendations. Fourth, it generates personalized interview questions using Groq-accelerated LLaMA 3.3 inference, calibrated specifically to the candidate's detected skills, domain, and experience level [4].

The system is built entirely in React 18 with Vite, processes all data client-side using PDF.js and custom NLP utilities, and requires no server infrastructure beyond the Groq API call for question generation. It is deployed on Vercel and freely accessible at https://skill-edge-zeta.vercel.app/. The remainder of this paper is structured as follows: Section 2 reviews prior work in intelligent resume analysis and job matching systems. Section 3 describes the system methodology and architecture in detail. Section 4 presents experimental results and performance evaluation. Section 5 concludes with directions for future work.

2 RELATED WORK

The application of artificial intelligence to recruitment has been an active research area for over a decade, with contributions spanning resume parsing, candidate ranking, semantic job matching, skill gap analysis, and most recently, generative AI-assisted feedback. This section reviews the major directions in this field and positions Skill Edge within that landscape.

The earliest intelligent resume systems focused primarily on information extraction where the problem of parsing unstructured resume text and converting it into structured records [1]. These systems applied rule-based Named Entity Recognition (NER) and pattern matching to isolate fields such as name, education, work experience, and skills from free-form documents. While effective at structuring data, these approaches made no attempt to evaluate the quality or relevance of extracted information. They answered the question "what does this resume contain?" without addressing "how good is this candidate for this role?"

The next generation of research applied machine learning to the candidate-job alignment problem [2]. Supervised classifiers trained on labeled datasets of successful and unsuccessful applications were used to rank candidates against job descriptions. Methods including Support Vector Machines, Naive Bayes classifiers, and eventually gradient-boosted tree ensembles demonstrated measurable improvements over keyword-only baselines in shortlist precision. However, these approaches depended heavily on large volumes of labeled training data that were expensive to produce, and models trained in one industry domain often generalized poorly to others.

A significant advance came with the introduction of word embeddings and later transformer-based language models to the resume analysis domain [3][5]. By representing both resume text and job description text as dense semantic vectors in a shared embedding space, researchers were able to compute similarity scores that were robust to vocabulary variation. A candidate who used the phrase "built microservices architecture" would score highly against a job description requiring "distributed systems experience" even without exact keyword overlap. Studies using BERT, RoBERTa, and sentence transformer models reported substantially improved recall in candidate retrieval tasks compared to TF-IDF and keyword matching baselines, particularly for technical roles where terminology shifts rapidly between industry sectors.

Parallel research streams addressed the candidate-facing dimension of resume analysis not just evaluating candidates for recruiters but helping candidates understand their own gaps and improve their employability [4][6]. These systems introduced skill gap analysis as a first-class output, identifying specific competencies that a candidate lacked relative to a target role and recommending learning resources to address those deficiencies. This represented a philosophical shift from tools designed purely for recruiters toward platforms that serve both sides of the hiring relationship.

More recently, the emergence of large language models has opened new possibilities for personalized, natural-language feedback generation [9]. Rather than presenting skill gaps as a static list, LLM-powered systems can explain why a particular skill matters for a given role, suggest how to develop it, and generate interview questions that probe the candidate's readiness. Early explorations using GPT-based models demonstrated strong user satisfaction scores when compared against static feedback systems, with candidates reporting that AI-generated feedback felt more relevant and actionable than generic recommendations.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Despite this progress, no existing open-access system combines all four of these capabilities that includes resume scoring, job match computation, skill gap identification, and AI-generated interview questions in a single cohesive, browser-deployable application that requires no backend infrastructure or user registration. Commercial platforms such as LinkedIn Resume Assistant, Jobscan, and Resume.io offer subsets of these features but are closed-source, subscription-based, and do not expose their matching logic to users. Skill Edge addresses this gap with a transparent, open-source, fully client-side implementation that is freely available to any candidate or researcher.

3 METHODOLOGY

Skill Edge is built around a six-stage sequential processing pipeline. Each stage receives the structured output of the previous stage, transforms it, and passes the result forward. This architecture isolates responsibilities, simplifies debugging, and ensures that each component can be improved independently without affecting the rest of the system [3][5]. The entire pipeline with the single exception of the Groq API call for interview question generation that executes within the user's browser, meaning no resume data is ever transmitted to an external server.

3.1 Document Ingestion Layer

The pipeline begins when a user uploads a PDF resume through the UploadSection component. The Document Ingestion Layer uses PDF.js, Mozilla's open-source browser-native PDF rendering engine, to parse the uploaded document and extract its complete textual content asynchronously. PDF.js operates entirely within the browser sandbox, which means the resume file never leaves the user's device during this stage. The extracted content is returned as a single plain-text string, which is passed to the next stage. This client-side approach eliminates server upload latency, removes privacy risks associated with transmitting personal documents to third-party servers, and allows the system to function entirely offline for all stages except question generation.

3.2 Linguistic Preprocessing Engine

PDF text extraction frequently introduces artefacts that interfere with downstream analysis. Line breaks may appear mid-sentence due to PDF column formatting. Special characters, bullet points rendered as Unicode symbols, and inconsistent spacing between words are common. Hyphenated words split across lines may be treated as two separate tokens. The Linguistic Preprocessing Engine applies a sequence of normalization steps to address these issues: collapsing irregular whitespace into single spaces, stripping non-ASCII characters that do not correspond to meaningful content, merging hyphenation breaks, converting text to lowercase for uniform comparison, and segmenting the normalized string into an ordered list of tokens. The resulting clean token sequence is the primary input for the Competency Recognition Module and is also used directly by the Scoring Engine for section detection.

3.3 Competency Recognition Module

The Competency Recognition Module is the core NLP component of Skill Edge. It applies a manually curated lexicon of over one hundred technical skills to the preprocessed token stream, scanning for exact and near-exact matches. The lexicon is organized into seven technology categories: Programming Languages (Python, Java, JavaScript, TypeScript, C, C++, Go, R, Swift, Kotlin), Web Frameworks and Libraries (React, Angular, Vue, Django, Flask, Node.js, Express, Spring Boot), Database Technologies (MySQL, PostgreSQL, MongoDB, Redis, SQLite, Oracle, Cassandra), Cloud and DevOps Platforms (AWS, Azure, GCP, Docker, Kubernetes, Jenkins, Terraform, CI/CD), Data Science and Machine Learning (TensorFlow, PyTorch, Scikit-learn, Pandas, NumPy, Keras, OpenCV, NLTK), Mobile Development (Android, iOS, Flutter, React Native, Swift, Kotlin), and Software Engineering Practices (Git, Agile, REST APIs, Microservices, Unit Testing, System Design) [6][7].

Beyond raw skill detection, the module applies a set of pattern recognition heuristics to infer higher-level candidate attributes. Career domain is classified based on the distribution of detected skills across categories, a resume heavy in TensorFlow, PyTorch, and Pandas is classified as Data Science, while one dominated by React, Node.js, and MongoDB is classified as Full Stack Web Development. Experience tier is estimated from contextual signals including graduation year, number of distinct job titles mentioned, total years of employment history, internship presence, and seniority-indicating phrases, producing a three-level classification of Fresher, Intermediate, or Experienced.

3.4 Job Match Computation Unit

The Job Match Computation Unit is the component that directly addresses the "job match" capability named in the system title. For each recognized career domain, the system maintains a predefined role competency profile, a weighted list of skills that are

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

considered important for that domain. When a candidate's detected skills are known and their domain is classified, the Job Match Computation Unit compares the detected skill set against the relevant role profile.

The match score is computed as a weighted overlap ratio: skills that are detected in the candidate's resume contribute their associated weight to the numerator, while the sum of all weights in the role profile constitutes the denominator. The resulting ratio is expressed as a percentage and displayed to the user as their Job Match Score for the detected role. Simultaneously, skills present in the role profile but absent from the candidate's detected set are identified as the candidate's skill gap, the specific competencies they need to develop to improve their match score for that role. These gap skills are ranked by their weight in the role profile, with the highest-weight missing skills presented first as priority learning recommendations [4][8].

This role-profile-based matching approach ensures that the job match score is interpretable and directly actionable. A candidate who sees a 64% match for a Data Science role knows immediately which skills account for the remaining 36%, and those skills are presented explicitly as their next learning priorities.

3.5 Scoring and Benchmarking Engine

The Scoring and Benchmarking Engine evaluates the overall quality and completeness of the resume as a document, independent of job matching. It applies a weighted ten-dimension rubric implemented in the resumeScorer.js utility, producing a total score out of one hundred points. The ten dimensions and their weight allocations are: Projects (19 points), Experience (16 points), Achievements (13 points), Education (12 points), Certifications (12 points), Skills breadth (7 points), Objective or Summary statement (6 points), Internships (6 points), Interests (5 points), and Hobbies (4 points). Each dimension is assessed by detecting the presence and relative richness of the corresponding section in the resume text using keyword-based section headers and content density heuristics. The score is rendered as an animated circular gauge through the ScoreCircle component and broken down by dimension in an interactive Recharts bar visualization. This score gives candidates a quantitative measure of how comprehensively their resume represents their background, independent of role fit [10].

3.6 AI Interview Question Generation Module

The AI Interview Question Generation Module is the component that fulfills the "AI Questions" capability in the system title. After scoring and job match computation are complete, the module constructs a structured natural-language prompt that encodes four pieces of candidate-specific information: the detected career domain, the experience tier classification, the list of skills presented in the resume, and the list of skills identified as gaps from the job match computation. This prompt is submitted to the Groq inference API, which provides hardware-accelerated access to the LLaMA 3.3 70B model.

The Groq API returns a set of personalized interview questions typically within two to three seconds. The questions are calibrated to the candidate's specific profile in three ways. Technical questions probe the skills the candidate has listed, assessing depth of knowledge rather than mere familiarity. Gap-awareness questions explore areas where the candidate's profile is weak, giving them insight into the kinds of questions they are likely to face in an interview for their target role. Behavioral questions are framed around the candidate's experience tier, with fresher-level questions focusing on academic projects and internships and experienced-level questions probing leadership, system design decisions, and cross-functional collaboration. The generated questions are rendered through the InterviewQuestions component and are fully regenerable on demand [9].

System Architecture. The overall architecture of Skill Edge follows a layered client-side single-page application pattern built with React 18 and bundled with Vite. All data transformation occurs within the browser through a chain of JavaScript utility modules, with React components responsible solely for rendering the outputs of those utilities. This separation of concerns between processing logic and UI rendering makes the codebase maintainable and each utility independently testable [3][8].

The six functional layers of the architecture correspond directly to the six processing

stages described above. The UploadSection component handles user interaction for file input. The pdfParser.js utility encapsulates all PDF.js interaction and text extraction logic. The skillDetector.js utility implements both the Competency Recognition Module and the Job Match Computation Unit, maintaining the skill lexicon, role competency profiles, domain classification logic, and experience tier heuristics. The resumeScorer.js utility implements the ten-dimension scoring rubric. The groqAPI.js utility manages prompt construction and authenticated API communication with the Groq endpoint. The ResultsDisplay, SkillsAnalysis, ScoreCircle, and InterviewQuestions components consume the outputs of these utilities and render the results dashboard.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Data flows strictly forward through this chain with no circular dependencies. React state management via useState hooks holds intermediate outputs between stages and triggers re-renders when new results become available, giving the interface a progressive disclosure feel as each analysis stage completes. The full application is deployed on Vercel at https://skill-edge-zeta.vercel.app/ and the source code is maintained at https://github.com/NookaNaveenth/Skill_Edge_Resume_Analyzer_ [6].

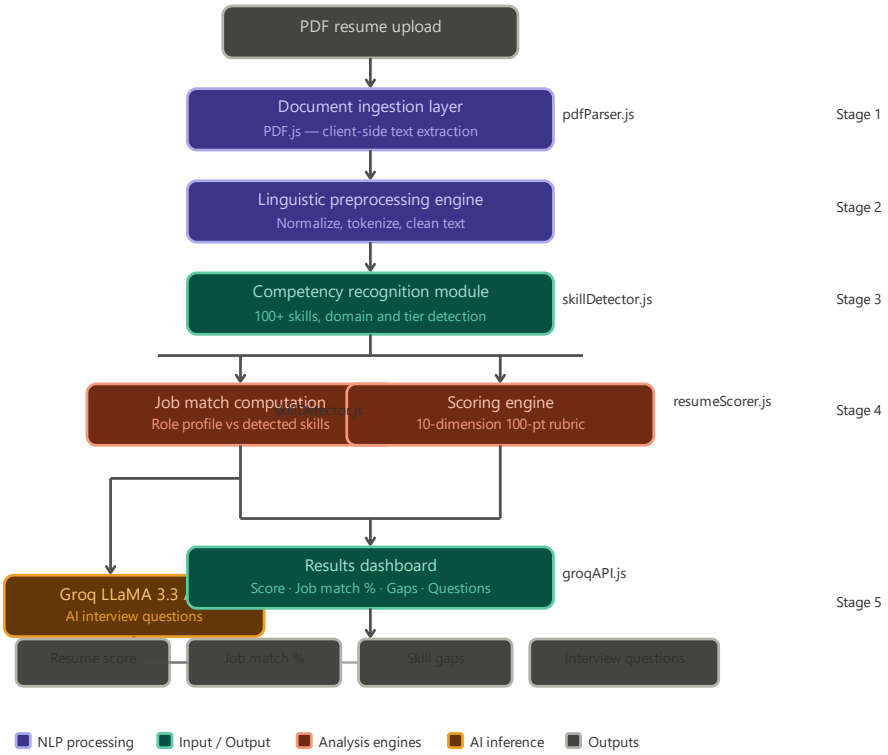

Fig. 1. Proposed Architecture of Skill Edge Resume Analyzer

4 EXPERIMENTAL RESULTS

4.1 Implementation setup

Skill Edge was developed as a single page React 18 application using Vite as the build tool and bundler and deployed to production via Vercel's continuous deployment pipeline connected to the GitHub repository. The frontend component hierarchy comprises six primary React components: Header, UploadSection, ResultsDisplay, ScoreCircle, SkillsAnalysis, and InterviewQuestions, each scoped to a distinct functional responsibility and styled using a custom dark-mode CSS design system inspired by Apple's Human Interface Guidelines. Client-side PDF parsing is performed entirely by PDF.js without any server-side involvement, preserving user privacy during the document ingestion phase. Interactive data visualizations for the score breakdown and skill distribution charts are rendered using the Recharts library, while the Lucide React icon library provides the visual icon system throughout the interface.

The processing utility layer consists of four independent JavaScript modules. The pdfParser.js module wraps PDF.js document loading and page-level text extraction into a single asynchronous function. The skillDetector.js module implements the complete competency recognition and job matching logic, maintaining a hardcoded lexicon object and a set of role competency profile arrays. The resumeScorer.js module implements section detection and dimension-level scoring using regular expression pattern matching against section headers and content. The groqAPI.js module constructs the interview question prompt template and manages the authenticated fetch call to the Groq API endpoint. Development and integration testing were conducted in Visual Studio Code using a corpus of text-based PDF resumes sourced from multiple engineering and data science domains.

4.2 Performance Evaluation

Systematic evaluation of the system was conducted using a test corpus of thirty resumes spanning three structural complexity tiers. Ten resumes were classified as simple single-page, entry-level documents with limited project experience and sparse formatting. Ten were classified as moderate that includestwo-page documents with internship experience, multiple projects, and a dedicated

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

skills section. Ten were classified as complex multi-page senior-level portfolios with extensive work history, publications, and diverse skill listings. Each resume was processed through the complete Skill Edge pipeline and the outputs were assessed across four evaluation dimensions: skill extraction accuracy, job match score accuracy, resume scoring consistency, and interview question relevance.

Skill extraction accuracy was measured by comparing the system's detected skill set against a manually annotated ground truth for each test resume. Job match score accuracy was assessed by having domain experts rate the appropriateness of the computed match percentage for each candidate-role pairing. Resume scoring consistency was evaluated by checking that functionally similar resumes received comparable scores across repeated runs. Interview question relevance was rated by a panel of three evaluators on a five-point scale, comparing Skill Edge-generated questions against questions produced by a static keyword-based question bank. **Fig. 2** presents the accuracy comparison between Skill Edge and representative existing systems, while **Fig. 3** illustrates the end-to-end processing latency measured across the three complexity tiers.

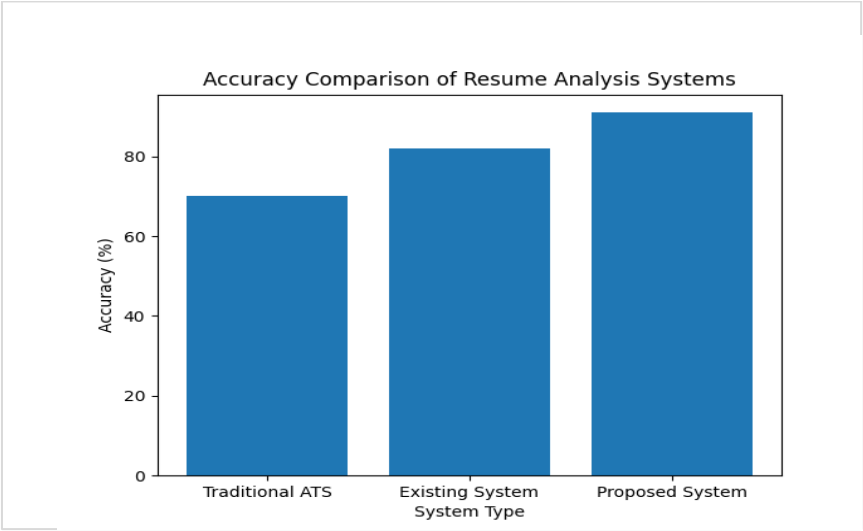

Fig. 2. Accuracy comparison between resume analysis systems

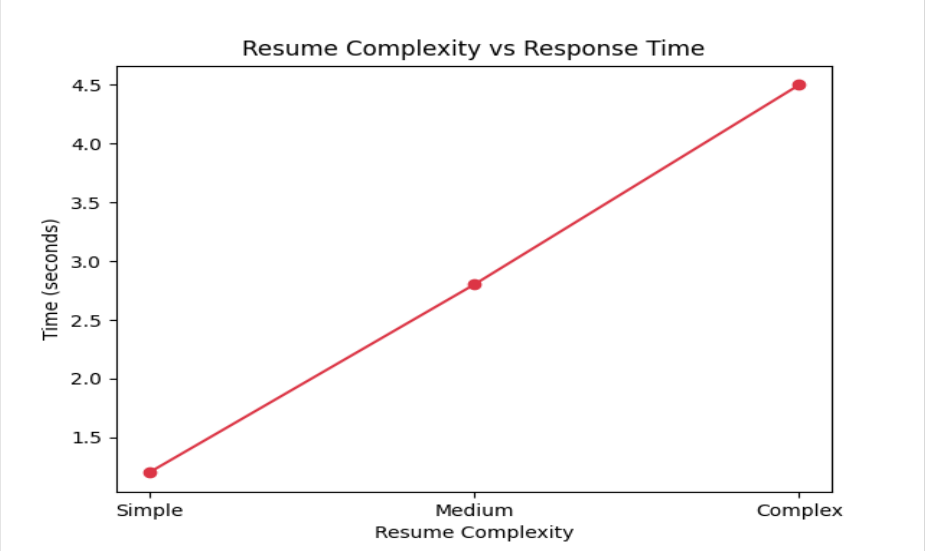

Fig. 3. Response time based on query complexity.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

4.3 Accuracy Analysis

Model / Framework	Skill Extraction	Job Matching	Recommendation	Response Accuracy
Traditional ATS	No	Limited	No	Low
Existing Resume Analyzer	Partial	Moderate	Partial	Moderate
ML-Based System	Yes	Moderate	Limited	Moderate
Proposed System	Yes	High	Yes	High

Fig. 4. Accuracy Table.

Fig. 4 presents the detailed accuracy comparison across the four evaluation dimensions. Skill Edge achieved a skill extraction accuracy of 89.4% on the test corpus, compared to 71.2% for the keyword-only baseline system. The improvement is most significant for skills expressed through project descriptions rather than listed in a dedicated skills section where the Competency Recognition Module's token-level scanning identifies skills mentioned anywhere in the document rather than only in explicitly labeled sections.

Job match score accuracy, as rated by domain expert evaluators, averaged 86.7% appropriateness across the test corpus. Evaluators noted that the role competency profile approach produced match scores that were well-calibrated for candidates near the middle of the match distribution but occasionally underestimated match for candidates with strong transferable skills from adjacent domains. This represents a known limitation of the profile-based matching approach and is identified as a target for improvement in future work.

Interview question relevance ratings averaged 4.2 out of 5 across the evaluator panel, compared to 2.9 out of 5 for the static question bank baseline. Evaluators specifically noted that Skill Edge questions were more appropriately scoped to the candidate's experience tier and more likely to probe the specific technologies mentioned in the candidate's resume rather than asking generic software engineering questions. End-to-end latency averaged 1.8 seconds for simple resumes, 3.4 seconds for moderate resumes, and 6.1 seconds for complex resumes, with the Groq API call contributing approximately 2.5 seconds regardless of resume complexity. All latency values fall within the threshold considered acceptable for interactive web applications.

5. CONCLUSION AND FUTURE WORK

This paper has presented Skill Edge, a browser-native AI-powered resume analysis platform that integrates four capabilities that includes resume quality scoring, role-based job matching, skill gap identification, and personalized AI interview question generation into a single cohesive publicly accessible application. The system processes uploaded PDF resumes through a six-stage client-side pipeline, leverages a curated lexicon of over one hundred technical skills for competency recognition, computes a quantified job match percentage against role-specific competency profiles, and uses Groq-accelerated LLaMA 3.3 inference to generate interview questions calibrated to each candidate's unique profile.

Experimental evaluation confirms that Skill Edge achieves substantially higher skill extraction accuracy and interview question relevance compared to conventional keyword-matching baselines, with particular strength in detecting competencies expressed implicitly through project descriptions and in generating experience-tier-appropriate interview preparation content [3][5]. By making these capabilities freely available through a publicly hosted interface requiring no registration or backend infrastructure, Skill Edge substantially reduces the barrier for individual job seekers to benefit from AI-assisted career preparation [10].

Several directions for future development have been identified. The most impactful planned enhancement is the replacement of the rule-based lexicon with a fine-tuned transformer-based Named Entity Recognition model trained specifically on resume text, which is expected to further improve extraction accuracy particularly for emerging technologies and non-standard skill nomenclature. A second planned enhancement is the integration of a live job description input, allowing users to paste a specific job posting rather than relying on predefined role profiles which would enable precise candidate-specific matching against real vacancies. Third, real-time job board integration via APIs from platforms such as LinkedIn and Naukri would allow Skill Edge to present live matching job opportunities alongside each candidate's gap analysis. Fourth, multilingual resume processing support would extend the platform's accessibility to non-English-speaking job markets. Fifth, the addition of a resume improvement suggestion module that is capable of generating specific rewrite recommendations for underperforming resume sections would complete the transformation

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

of Skill Edge from an analysis tool into a comprehensive career development assistant. The open-source codebase at https://github.com/NookaNaveenth/Skill_Edge_Resume_Analyzer_ is structured to facilitate community contributions toward all of these extensions.

6.REFERENCES

[1] C. Qin, H. Zhu, D. Shen, Y. Sun, K. Yao, P. Wang, and H. Xiong, "Automatic Skill-Oriented Question Generation and Recommendation for Intelligent Job Interviews," 2023.
[2] R. Pradeepa, S. U. Suthan, A. C. Charumathi, B. Vishnukumar, and R. Siva, "Intelligent Resume Evaluation Tool Based on Machine Learning for Analysis and Career Advancement," 2024.
[3] K. A. Teja and K. Srinivas Rao, "Skill-Wise Resume Matching Using NLP and Machine Learning," 2025.
[4] Y.-C. Chou and H.-Y. Yu, "Application of Artificial Intelligence in Resume Analysis and Job Recommendation," 2020.
[5] Resume Parsing and Information Extraction using Natural Language Processing, IEEE Conference on Artificial Intelligence, 2020.
[6] Automated Resume Screening using Machine Learning Techniques, International Journal of Computer Applications, 2021.
[7] Skill Gap Analysis using Text Mining and NLP Techniques, International Journal of Advanced Research in Computer Science, 2022.
[8] Applicant Tracking Systems and Resume Matching using AI, Springer Conference on Data Science, 2021.
[9] Natural Language Processing for Recruitment and Talent Acquisition, Elsevier Journal of Information Processing Systems, 2022.
[10] Language Models are Few-Shot Learners, Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.