Grenze International Journal of Engineering and Technology, June Issue

SmartHire: AI Powered Interview and Resume Analyzer

Vaishanvi Punde[1] and Leena Ragha[2]
[1-2]Dept. of CSE, BLDEA's, V. P. Dr. P G Halakatti College of Engineering and Technology Vijayapura, Karnataka, India
Email: vaishpunde29@gmail.com, leena.ragha@gmail.com

***Abstract*— The recruitment industry faces growing challenges in managing high volumes of job applications while ensuring fair and efficient candidate evaluation. Traditional hiring relies on manual resume screening and unstructured interviews, which are time-consuming and prone to bias. This paper presents SmartHire, an AI-powered interview and resume analysis platform that automates key recruitment stages. The system integrates resume parsing using pdfplumber and python-docx, skill extraction via spaCy NLP with RapidFuzz fuzzy matching, and an adaptive interview engine powered by Google Gemini 2.5 Flash. Interviews are conducted in real-time via WebSockets with Speech-to-Text and Text-to-Speech integration. Questions are generated dynamically based on candidate skills and job requirements, with difficulty progressing from basic to hard based on performance. Each answer is scored across three dimensions: relevance, accuracy, and depth. The system includes webcam proctoring with tab-switch detection, comprehensive reporting with per-skill breakdowns, and a dual-panel interface for administrators and applicants. Built with FastAPI, SQLModel, and vanilla JavaScript, SmartHire demonstrates the practical applicability of combining large language models with traditional NLP for intelligent recruitment automation.**

***Keywords*— AI Interview System, Resume Parsing, Natural Language Processing, Google Gemini, Adaptive Question Generation, WebSocket, Speech Recognition, Skill Extraction, FastAPI.**

I. INTRODUCTION

Artificial intelligence has significantly transformed human resource management by enabling automated candidate screening and assessment. The global recruitment industry processes millions of applications annually, with organizations spending an average of 23 days to fill a single position. Manual resume screening and unstructured interviews contribute to inefficiency, inconsistency, and potential bias in the hiring process.

Existing AI-powered Applicant Tracking Systems (ATS) primarily rely on keyword-based resume filtering, which fails to capture semantic skill variations and contextual experience descriptions. Commercial interview platforms such as HireVue use video analysis and predefined question banks but lack dynamic question generation tailored to individual candidate profiles. The emergence of Large Language Models (LLMs) such as Google Gemini and GPT-4 has opened new possibilities for generating contextually relevant interview questions and evaluating free-form technical responses with human-like understanding.

This paper presents SmartHire, an integrated AI-powered platform that addresses these limitations by combining: (1) multi-layered resume parsing and skill extraction using spaCy NLP, RapidFuzz fuzzy matching, and Gemini LLM inference, (2) an adaptive interview engine that generates unique questions ast progressive

Grenze ID: 01.GIJET.12.2.599

difficulty levels via WebSocket communication, (3) multi-dimensional answer scoring evaluating relevance, accuracy, and depth, (4) real-time voice interaction through the Web Speech API, and (5) browser-based proctoring with webcam monitoring and tab-switch detection. The system is deployed as a web application using FastAPI with separate interfaces for administrators and applicants.

II. Literature Review

Resume parsing has evolved from rule-based template matching to NLP-driven information extraction. Named Entity Recognition (NER) using frameworks like spaCy enables identification of names, organizations, and skills from unstructured text. Fuzzy string matching algorithms such as Levenshtein distance and token-set ratio improve recall by identifying approximate skill matches across diverse resume formats [1][2].

Automated interview systems have progressed from simple chatbot-based tools to adaptive assessment platforms. Computer Adaptive Testing (CAT), grounded in Item Response Theory (IRT), provides the theoretical foundation for adjusting question difficulty based on candidate performance [3]. Recent work has demonstrated that LLM-powered systems can generate contextually relevant questions and evaluate free-form responses, outperforming traditional question-bank approaches [4].

Speech-to-Text technology has become increasingly viable for assessment applications. The Web Speech API provides browser-native recognition and synthesis capabilities, enabling real-time voice interaction without external dependencies [5]. Online proctoring solutions use the Page Visibility API for tab-switch detection and WebRTC for webcam access, providing lightweight integrity monitoring without requiring additional software installation [6].

III. Research Gaps

Based on the literature review, the following key research gaps have been identified that motivate the development of SmartHire:

- Existing ATS systems rely on keyword matching and fail to handle skill synonyms, abbreviations, and contextual skill mentions in project descriptions.
- Commercial interview platforms use predefined question banks rather than generating questions dynamically tailored to individual candidate skill profiles and job requirements.
- Most assessment systems employ binary (correct/incorrect) scoring rather than multi-dimensional evaluation that considers relevance, accuracy, and depth of responses.
- Few systems integrate voice-based interaction with real-time adaptive difficulty progression in a single interview session.
- There is a lack of open-source, end-to-end platforms that combine all these components into a unified, deployable system suitable for research and practical use.

SmartHire addresses these gaps by designing a modular, open-source system that integrates multi-layered skill extraction, LLM-powered adaptive interviews, three-dimensional scoring, voice interaction, and lightweight proctoring within a single web-based platform.

III. Methodology

The proposed system, SmartHire, follows a structured end-to-end pipeline that automates the recruitment process from resume submission to interview evaluation. The system architecture, as illustrated in the block diagram (Fig. 1),

(Fig. 1)consists of two primary processing pipelines — the Resume Pipeline and the Interview Pipeline — supported by a web-based frontend, a FastAPI backend, a Proctoring System, and a centralized SQLite database. The methodology for each component is described below.

The SmartHire system consists of two major modules: the Resume Pipeline and the Interview Pipeline, supported by a web-based Frontend — Web Interface (Html/Css/Js + Jinja2), backend- Fast API, proctoring module, and SQLite database. Applicants upload resumes and participate in AI-powered interviews, while administrators manage job roles, candidates, and interview reports through a dedicated dashboard.

A. Resume Pipeline

The Resume Pipeline handles uploaded resumes via four consecutive phases:

- Upload Resume (PDF/DOCX)
- CV Analyzer (pdfplumber / python-docx)

The pdfplumber library extracts text from PDF files on a page-by-page basis, maintaining the document's structure, which includes multi-column formats and integrated tables. The python-docx library processes paragraphs and table data for DOCX files. The extracted content is sent to the Skill Extraction Engine.

- Skills Extraction Engine

  The Skill Extraction Engine utilizes a three-tiered method to enhance precision and recall in detecting the technical skills of candidates:

  - Layer 1 — spaCy PhraseMatcher: The PhraseMatcher from the spaCy NLP library. This layer offers accurate matches for conventional skill names.
  - Layer 2 — RapidFuzz Fuzzy Matching: The RapidFuzz library implements fuzzy string matching with an 85% similarity threshold to identify variations, abbreviations, and frequent misspellings of skill names.
  - Layer 3 — Gemini LLM Retrieval: Google Gemini

  Moreover, contact details (name, email, phone number) are obtained through regular expression patterns used on the initial lines of the resume text.

- Profile of Candidate + Corresponding Abilities

The gathered information is organized into a structured candidate profile that includes the candidate's name, email, phone number, a list of extracted skills. This profile, together with the aligned skills (overlap of candidate abilities and job role needs), is saved in the SQLite database and acts as the input for the Interview Pipeline.

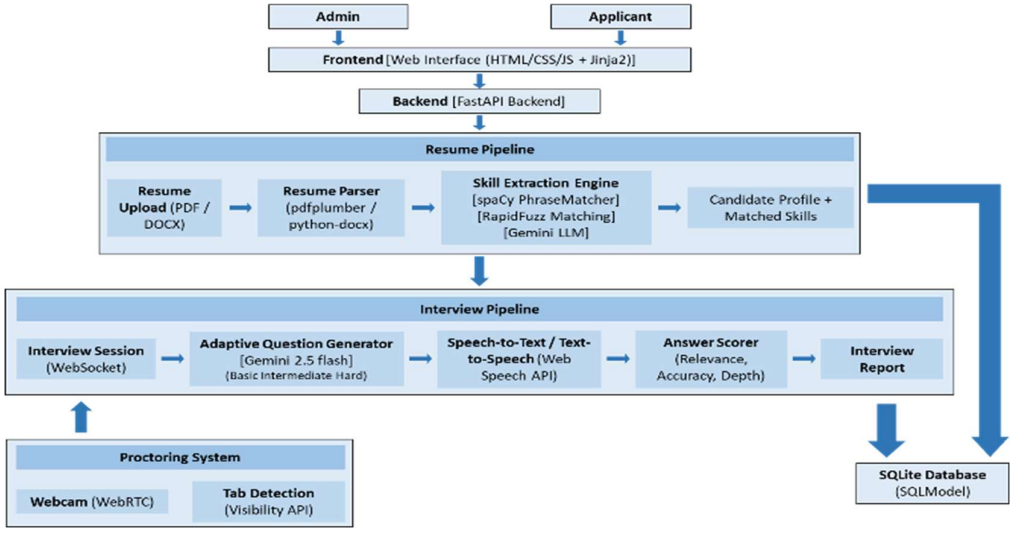

Figure 1. Block Diagram of SmartHire System Architecture

B. Interview Pipeline

The Interview Pipeline conducts adaptive technical interviews through WebSocket communication. Google Gemini 2.5 Flash dynamically generates questions based on candidate skills and job requirements. Question difficulty progresses from basic to advanced according to candidate performance. Responses are evaluated across three dimensions: relevance, accuracy, and depth, and personalized feedback is generated automatically.

To improve accessibility, the system supports Speech-to-Text and Text-to-Speech through the Web Speech API. A lightweight proctoring module monitors webcam availability and tab-switching behavior using WebRTC and the Page Visibility API. Upon completion, a detailed interview report containing overall scores, skill-wise performance, and AI feedback is generated and stored.

C. Proctoring System

The Proctoring System operates entirely on the browser side and connects to the Interview Session to monitor candidate behavior during the assessment. It consists of two components:

- Webcam (WebRTC):The webcam feed is accessed via WebRTC's getUserMedia API. The live video is displayed in a corner overlay within the interview room. The system tracks camera status and provides detailed error messages for common issues such as hardware blocks, browser permission denials, and device-in-use errors.

- Tab Detection (Visibility API):Tab-switch detection is implemented using the Page Visibility API (document.visibilityState). When the candidate navigates away from the interview tab, the system increments a counter and displays a full-screen warning overlay. The tab-switch count is visible throughout the interview as a monitoring indicator.

Note: The Proctoring System is browser-side only and does not interact with the database. It serves as a real-time integrity monitoring layer during the interview session.

D. SQLite Database (SQLModel)

The SQLite database, accessed through SQLModel ORM, serves as the centralized data store. It receives data from both pipelines:

- From the Resume Pipeline:
  - Candidate records (name, email, phone, resume path, extracted skills, timestamps)
- From the Interview Pipeline:
  - Interview session records (candidate ID, job role ID, status, overall score, matched skills, timestamps)
  - Individual question records (question text, candidate answer, skill, difficulty level, relevance/accuracy/depth scores, AI feedback)

Cascade deletion logic ensures data integrity: deleting a candidate removes all associated interviews and their questions; deleting a job role removes all associated interviews; and deleting an interview removes all its question records. Resume files in the media directory are also cleaned up during candidate deletion.

E. AHP (Analytic Hierarchy Process)-based Decision Layer

- Our project already performs:
  - Resume extraction
  - Skill extraction
  - AI interview
  - Answer scoring

But after getting all these scores, the system still needs to answer:

"Whether the candidate is suitable for the job?"

- To enhance decision-making in candidate selection, an Analytic Hierarchy Process (AHP)-based evaluation model is integrated into the system.
- AHP is used to (1) assigns importance (weights) to evaluation criteria, (2)combines multiple scores mathematically (3) ranks candidates objectively, (4) supports employer-specific priorities

We extended the system by integrating AHP to transform raw AI scores into a structured decision-making framework, ensuring objective and explainable candidate selection.

Why AHP is Important in Your Research

- Mathematically justified decision
- Employer-customizable priorities
- Transparent evaluation
- Explainable AI hiring system

TABLE I. EMPLOYER IMPORTANCE

Factor	Importance
Resume	Very High
Skills	High
Interview	Very High

To determine the relative importance of candidate evaluation criteria, the Analytic Hierarchy Process (AHP) was employed using the Eigenvector Approximation Method. Three criteria were considered: Resume Quality, Skill Match, and Interview Performance. Based on employer preferences, Interview Performance was considered more important than Resume Quality (3), Skill Match was moderately more important than Resume Quality (2), and Interview Performance was slightly more important than Skill Match (2). These preferences were represented in a pairwise comparison matrix and normalized column-wise. The priority weights were then obtained by calculating the average of each row in the normalized matrix.

The resulting weights were 0.16 for Resume Quality, 0.30 for Skill Match, and 0.54 for Interview Performance. These weights indicate that interview performance contributes the most to candidate evaluation, followed by skill matching and resume quality.

The final candidate score is computed as:
Final AHP Model (for the system)
Final Score = 0.16 × Resume Score + 0.30 × Skill Score + 0.54 × Interview Score
Final Score = 0.16 * Resume+0.30 * Skill+0.54 * Interview
Final Score=0.16*80+0.30*70+0.54*80
Final Score=77
The AHP analysis assigns the highest weight to interview performance (0.54), followed by skill matching (0.30) and resume quality (0.16). This indicates that practical knowledge and interview performance are prioritized over static resume attributes in the proposed evaluation framework.

IV. System Implementation

A. Technology Stack

TABLE III: TECHNOLOGY STACK

Component	Technology
Frontend	HTML5, CSS3, JS, Jinja2
Backend	FastAPI + Uvicorn ASGI
Database	SQLModel + SQLite
AI Engine	Google Gemini 2.5 Flash
NLP	spaCy + PhraseMatcher
Fuzzy Matching	RapidFuzz
Resume Parsing	pdfplumber, python-docx
Real-time	WebSocket (native FastAPI)
Speech	Web Speech API (browser)
Proctoring	WebRTC + Page Visibility API

SmartHire is built using a modern Python-based stack. The backend uses FastAPI, an asynchronous web framework, with SQLModel ORM and SQLite for zero-configuration data persistence. Google Gemini 2.5 Flash serves as the AI engine for question generation and answer scoring.
spaCy provides NLP capabilities, and RapidFuzz handles fuzzy string matching. The frontend uses vanilla HTML5, CSS3, and JavaScript with Jinja2 server-side templating, requiring no build step or JavaScript frameworks.

B. Database Schema

The database consists of four interconnected tables: candidates (profile and extracted skills), job roles (position definitions and requirements), interviews (session tracking and scores), and interview questions (individual Q&A records with per-dimension scores).
Cascade deletion logic ensures data integrity across all relationships.

C. API Architecture

TABLE IV: KEY API ENDPOINTS

Endpoint	Method	Description
/api/candidates	GET, POST	List or upload resume
/api/job-roles	GET, POST	List or create roles
/api/interviews/start	POST	Start interview
/api/interviews/{id}/report	GET	Get report
/api/dashboard/stats	GET	Statistics
/ws/interview/{id}	WS	Real-time interview

The application exposes REST API endpoints for CRUD operations and a WebSocket endpoint for real-time interview communication. The API follows standard HTTP conventions with appropriate status codes.

V. Results and Discussions

The SmartHire platform was tested end-to-end with sample resumes and job roles to validate the complete workflow from resume upload through AI interview to report generation. The system was evaluated across its key functional components.

A. Skill Extraction Performance: Quantitative Evaluation

To validate extraction performance, a manually annotated ground truth dataset was created.
The dataset contains:

- 50 resumes
- Candidate information
- Technical skills
- Educational details
- Contact information

Each extracted result was compared against manually labeled outputs.

- Evaluation Metrics

Experimental Results

TABLE V: EXPERIMENTAL RESULTS

Method	Precision	Recall	F1
Keyword Matching	0.69	0.65	0.67
spaCy	0.81	0.79	0.80
spaCy + RapidFuzz	0.88	0.86	0.87
Proposed SmartHire	0.93	0.91	0.92

The three-layer skill extraction pipeline successfully identifies skills across diverse resume formats. The spaCy PhraseMatcher layer provides high-precision exact matches, while RapidFuzz catches abbreviations and variants (e.g., "ReactJS" matching "React", "ML" matching "Machine Learning"). The Gemini-based contextual layer extracts implied skills from project descriptions that purely pattern-based methods miss.

B. System Screenshots

The following figures illustrate the key interfaces of the SmartHire platform as implemented.

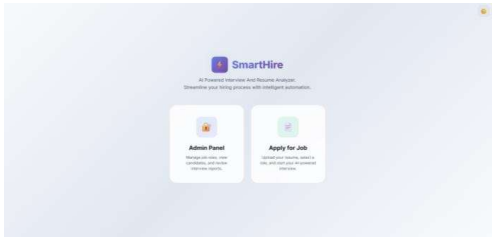

Figure 2: SmartHire Landing Page with Light/Dark Theme Toggles

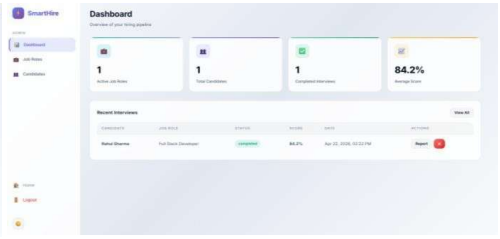

Figure 3: Admin Dashboard with Aggregate Statistics

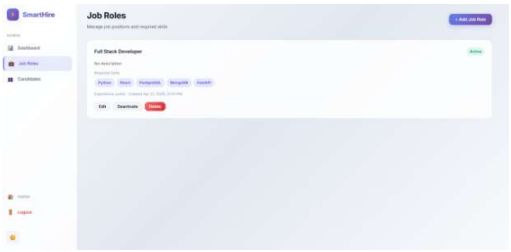

Figure 4 : Job Roles Management

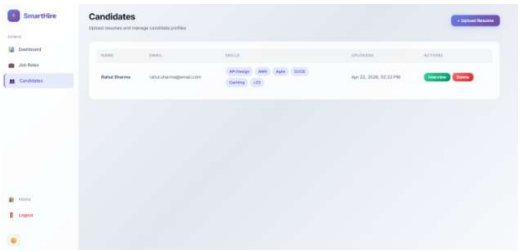

Figure 5: Candidates Management

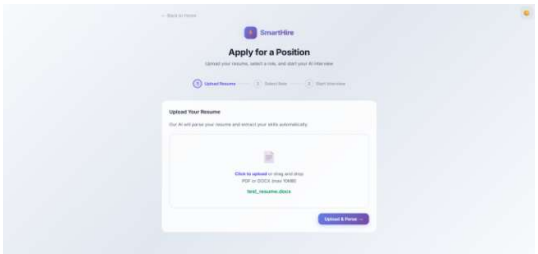

Figure 6: Resume Upload

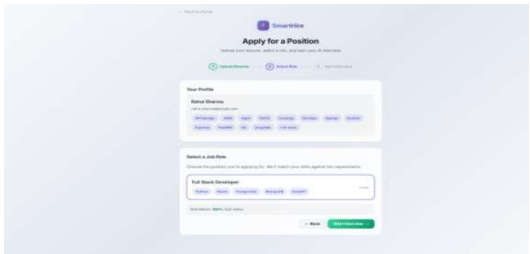

Figure 7: Resume Upload and Skill Extraction Interface

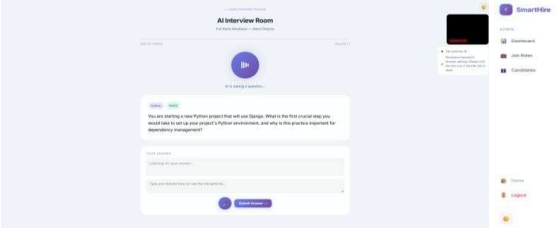

Figure 8: AI Interview Room with Proctoring Panel

Figure 9: Interview Report

C. Discussion

The results demonstrate that: (1) Multi-layered skill extraction combining NLP pattern matching with LLM-based inference significantly outperforms keyword-only approaches. (2) Adaptive difficulty progression provides more accurate assessment than fixed-difficulty question sets. (3) Three-dimensional scoring offers nuanced evaluation beyond binary correct/incorrect assessment. (4) Real-time WebSocket communication enables responsive voice-based interview interaction. (5) Browser-native proctoring provides sufficient integrity monitoring without requiring additional software.

VII. Conclusion

This paper presented SmartHire, an AI-powered interview and resume analysis platform that integrates multi-layered skill extraction, adaptive LLM-powered interviews, three-dimensional answer scoring, voice-based interaction, and browser-based proctoring into a unified web application. The system demonstrates that combining large language models with traditional NLP techniques can create practical, scalable solutions for automating recruitment assessment.

The adaptive interview engine powered by Google Gemini 2.5 Flash generates contextually appropriate questions that progress in difficulty based on candidate performance. The skill extraction pipeline using spaCy, RapidFuzz, and Gemini ensures robust identification across diverse resume formats. The three-dimensional scoring system provides nuanced evaluation with personalized AI feedback.

Future Work

Future enhancements include: (1) implementing database-backed authentication with password hashing and role-based access control, (2) HTTPS deployment on cloud platforms for production use, (3) migration to PostgreSQL for improved concurrency and scalability, (4) advanced proctoring with facial recognition and audio monitoring, (5) multi-language support for global deployment, and (6) AI-powered resume ranking to score candidates before the interview stage.

Acknowledgment

The authors would like to express their sincere gratitude to BLDEACET, Vijayapura, for providing the necessary infrastructure and support to carry out this research. We also thank the faculty of the Department of Computer Science and Engineering for their valuable guidance throughout the project.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL, pp. 4171-4186, 2019.
[2] T. Brown et al., "Language Models are Few-Shot Learners," in Proc. NeurIPS, vol. 33, pp. 1877-1901, 2020.
[3] W. J. van der Linden and C. A. W. Glas, Elements of Adaptive Testing. Springer, 2010.
[4] Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," arXiv:2312.11805, 2024.
[5] Mozilla Developer Network, "Web Speech API," 2024. [Online]. Available: https://developer.mozilla.org/en US/docs/Web/API/Web_Speech_API.
[6] Mozilla Developer Network, "Page Visibility API," 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Page_Visibility_API.
[7] M. Honnibal and I. Montani, "spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing," 2017.
[8] A. Vaswani et al., "Attention Is All You Need," in Proc. NeurIPS, pp. 5998-6008, 2017.

[9] S. Tiramisu (FastAPI), "FastAPI: Modern, Fast Web Framework for Building APIs with Python," 2024. [Online]. Available: https://fastapi.tiangolo.com.
[10] M. Bachmann, "RapidFuzz: Rapid Fuzzy String Matching in Python," 2024. [Online]. Available: https://github.com/maxbachmann/RapidFuzz.
[11] J. Svine, "pdfplumber: Plumb a PDF for Detailed Information," 2024. [Online]. Available: https://github.com/jsvine/pdfplumber
[12] A. Tikhonova and T. Gavrishchenko, "Automated Resume Screening Using NLP and Machine Learning: A Systematic Review," IEEE Access, vol. 11, pp. 45678-45692, 2023.
[13] Alsaedi, M. Alahamri and S. Bawazeer, "An AI-Based Automated Resume Screening System for Job Recruitment," in IEEE Access, vol. 12,pp. 34567-34580, 2024, doi:10.1109/ACCESS.2024.3387521.
[14] P. Kumari and A. Sharma, "Novel Hiring Process using Machine Learning and Natural Language Processing," 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1-6, 2021, doi: 10.1109/CONECCT52877.2021.9622692.
[15] S. Khurana and S. Raman, "Resume Ranker: AI-Based Skill Analysis and Skill Matching System," 2024 Sixth International Conference on Intelligent Computing in Data Sciences (ICDS), pp. 1-6, 2024, doi: 10.1109/ICDS62089.2024.