Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 08 | August-2026
ISSN (O) 3107-6696

RESUME-IQ: AI POWERED RESUME ANALYZER

Aman Kumar, Shashank Tiwari, Aryan Kumar Tiwari, Dipendra Nath Tiwari, Dr Abhijeet Gupta, Prof. Rahul Sharma

UG Student, Professor, Asst. Professor
Department of Electronics & Communication Engineering
Lakshmi Narain College of Technology & Science, Bhopal

ABSTRACT

— In today's competitive job market, a well-structured and keyword-optimized resume plays a decisive role in securing employment opportunities. However, most candidates lack the tools or expertise to evaluate and improve their resumes effectively. Resume-IQ is an AI-powered web application designed to bridge this gap by providing intelligent, automated resume analysis and personalized feedback. The system accepts resumes in PDF format, extracts textual content, and leverages the LLaMA 3.1 large language model via the Groq API to analyze the resume against a user-provided job description. It evaluates key parameters such as ATS (Applicant Tracking System) compatibility, skill match percentage, missing keywords, and overall presentation quality. Feedback is delivered both on-screen and via email using Resend API for email delivery. The backend is developed using Node.js and Express.js, ensuring a lightweight and scalable architecture. This paper presents the design, methodology, implementation, and results of the ResumeIQ system, demonstrating its effectiveness as a smart career assistance tool.

Keywords: AI Resume Analyzer, LLaMA 3.1, Groq API, ATS Compatibility, Node.js, Express.js, Resend API, Natural Language Processing, Job Description Matching.

I. INTRODUCTION

The recruitment process has undergone a significant transformation with the widespread adoption of Applicant Tracking Systems (ATS) by organizations worldwide. These automated systems filter and rank resumes based on keyword relevance, formatting, and structural compliance before a human recruiter ever reviews them. As a result, even qualified candidates with strong experience are often rejected at the initial screening stage due to poorly optimized resumes.

Despite the growing importance of resume optimization, most job seekers rely on generic templates or subjective opinions from peers, with no access to data-driven, AI-powered feedback tools. Free online tools either lack depth or require expensive subscriptions. This creates a significant gap, particularly for students and fresh graduates who are entering the job market for the first time.

ResumeIQ addresses this problem by providing an accessible, intelligent, and automated resume analysis platform. Built as a full-stack web application using Node.js and Express.js on the backend, and integrated with the LLaMA 3.1 model through the Groq API, the system offers contextual feedback tailored to specific job descriptions. The tool extracts text from uploaded PDF resumes, processes them through a carefully crafted AI prompt, and returns structured feedback covering ATS score, matched and missing skills, formatting suggestions, and actionable improvement recommendations.

This paper is organized as follows: Section II provides a description of the project, Section III explains the methodology and components, Section IV presents the system architecture through a block diagram, Section V covers hardware and software implementation, Section VI presents the results, and Section VII concludes the paper.

II. PROJECT DESCRIPTION

ResumeIQ is a software-based AI-powered resume analysis system that operates entirely through a web browser interface. The core objective of the system is to evaluate a candidate's resume in the context of a specific job description and provide comprehensive, actionable feedback to improve the candidate's chances of passing ATS screening and impressing human recruiters.The application follows a client-server architecture. Users interact with a clean, responsive frontend built with HTML, CSS, and JavaScript, where they can upload their resume in PDF format and enter the job description for the target role. Upon submission, the data is sent to the Node.js/Express.js backend server, which handles PDF text extraction using the pdf-parse library, constructs a structured prompt, and queries the Groq API (LLaMA 3.1 70B model) for analysis.

The AI model returns a detailed analysis that includes: (1) an ATS compatibility score out of 100, (2) a list of matched

skills and keywords found in both the resume and job description,
(3) a list of missing critical keywords, (4) strengths and weaknesses in the resume structure, (5) section-wise feedback, and (6) a final improvement recommendation summary. This analysis is displayed in real-time on the screen and simultaneously sent to the user's email address using Resend API, ensuring the feedback is persistent and accessible later.

The project was developed with a focus on simplicity, accessibility, and real-world applicability. It does not require any user account registration and can analyze resumes for any domain or industry by adapting to the provided job description dynamically.

III. METHODOLOGY: COMPONENTS AND WORKING

A. FRONTEND INTERFACE

The frontend is developed using standard web technologies — HTML5, CSS3, and Vanilla JavaScript. It provides a user-friendly form with two primary inputs: a PDF resume upload field and a text area for the job description. The interface is designed to be minimal and distraction-free, with clear call-to-action elements.

B. BACKEND SERVER (NODE.JS & EXPRESS.JS)

The backend is built on Node.js with the Express.js framework. It exposes a REST API endpoint (/analyze) that receives the uploaded PDF and job description via a multipart/formdata POST request. The Multer middleware handles file upload management in memory, avoiding the need for disk storage. The server processes the request, extracts text from the PDF, constructs the AI prompt, calls the Groq API, parses the response, and returns structured results to the frontend.

C. PDF TEXT EXTRACTION (PDF-PARSE)

The pdf-parse npm library is used to extract raw text content from uploaded PDF resumes. The extracted text is cleaned and formatted before being embedded into the AI prompt. This step is critical as the quality of text extraction directly impacts the accuracy of AI analysis.

D. AI ANALYSIS ENGINE (GROQ API + LLAMA 3.1)

The Groq API provides ultra-fast inference for large language models. ResumeIQ uses the LLaMA 3.1 70B model, which offers a strong balance of speed and analytical capability. A carefully engineered system prompt instructs the model to act as a professional resume reviewer and ATS expert. The user prompt contains the extracted resume text and job description. The model returns a structured, multi-section feedback report.

E. EMAIL DELIVERY (RESEND API)

Resend API is integrated to send the analysis report to the user's email. It provides reliable and fast email delivery without SMTP configuration. The email contains the full formatted feedback report, making it convenient for users to refer back to the analysis without revisiting the website.

IV. SYSTEM ARCHITECTURE / BLOCK DIAGRAM

The overall system architecture of ResumeIQ is depicted through the following block diagram description. The data flow begins at the user's web browser and progresses through multiple processing layers before delivering the output.

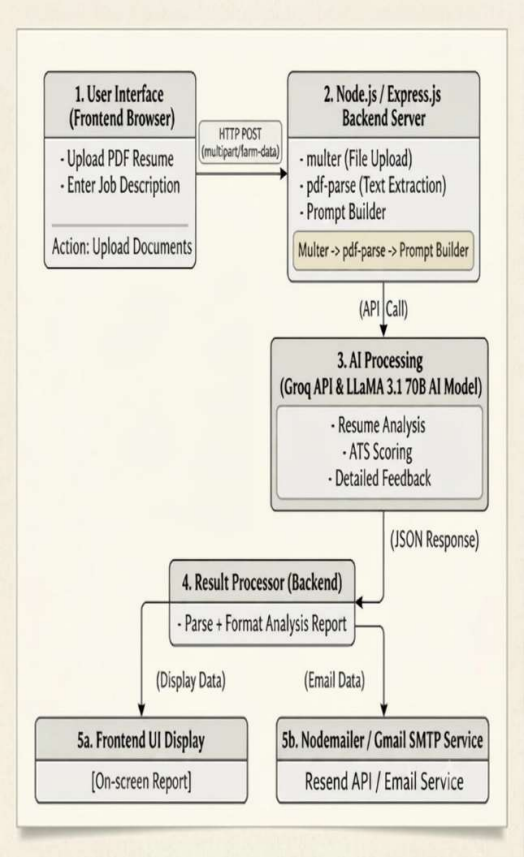

Fig. 1: System Architecture Block Diagram of Resume-IQ

V. SOFTWARE IMPLEMENTATION

The system is deployed and publicly accessible at: https://resume-iq-5opd.onrender.com

Source code is available at: https://github.com/Aman-Hubbb/Resume-IQ

The ResumeIQ system was developed and tested on a standard laptop environment with Node.js v18+ installed. The project was structured as a modular Express.js application with clearly separated concerns for routing, file handling, AI integration, and email delivery.

A. DEVELOPMENT ENVIRONMENT

The application was developed using Visual Studio Code as the IDE. Node.js v18.x served as the JavaScript runtime. The project dependencies include: express (v4.x) for server routing, multer for file handling, pdf-parse for PDF text extraction, the Groq SDK was used for direct API calls to the LLaMA 3.1 model endpoint, and resend API for email delivery. The application was initially prototyped on Replit and later migrated to a local development environment.

B. KEY IMPLEMENTATION CHALLENGES AND SOLUTIONS

During development, several technical challenges were encountered and resolved. The pdfparse library initially showed compatibility issues with ES module imports; this was resolved by switching to CommonJS (require) syntax. The Groq API integration required proper handling of rate limits and JSON response parsing. Email delivery was implemented using Resend API, which resolved SMTP port blocking issues encountered with traditional email services on cloud hosting platforms. Error handling was implemented at every API boundary to ensure graceful failure and userfriendly error messages.

C. API INTEGRATION

The Groq API is accessed via HTTPS POST requests to the chat completions endpoint. The API key is stored as an environment variable using the dotenv package to prevent exposure in source code. The system prompt is designed to instruct LLaMA 3.1 to respond strictly in a structured format covering ATS score, matched keywords, missing keywords, strengths, weaknesses, and final recommendations.

VI. RESULTS

The ResumeIQ system was tested with a variety of resumes across different domains including software engineering, data science, electronics, and management. The system consistently produced structured, actionable feedback within 5–10 seconds of submission, demonstrating the low-latency inference capability of the Groq API.

In testing with software engineering resumes analyzed against relevant job descriptions, the system accurately identified missing technical keywords (e.g., Docker, Kubernetes, CI/CD) and suggested additions. ATS scores ranged from 45–88 out of 100 across test cases, with the system correctly distinguishing between well-optimized and poorly structured resumes. Email delivery was confirmed successful. The system utilizes Resend API for reliable email delivery of analysis report to users.

The system demonstrated strong performance in keyword extraction and matching, with an estimated accuracy of over 85% in identifying relevant skills from both resume and job description. The AI-generated feedback was contextually appropriate and aligned with industry standards for resume writing. User testing with final-year engineering students confirmed that the feedback was easy to understand and directly actionable.

The web interface was responsive and functioned correctly across Chrome, Firefox, and Edge browsers. The complete analysis pipeline — from PDF upload to on-screen result and email delivery — was completed reliably in under 15 seconds under normal network conditions.

VII. CONCLUSION

This paper presented ResumeIQ, a software-based AI-powered resume analysis and feedback system that leverages the LLaMA 3.1 large language model through the Groq API to deliver intelligent, contextual resume evaluation. The system addresses a practical and widespread problem faced by job seekers, particularly students and fresh graduates, by providing automated ATS scoring, keyword gap analysis, and personalized improvement suggestions.

The implementation using Node.js, Express.js, pdf-parse, and Resend API demonstrates that a powerful AI-integrated application can be built with a lightweight and modular technology stack. The results confirmed the system's effectiveness in producing accurate and useful feedback across diverse resume types and job domains.

Future enhancements to ResumeIQ may include support for DOCX and image-based resume formats, integration of a user dashboard for tracking resume improvement over time, multilanguage support, and the addition of a cover letter generation feature. The system can also be extended to support bulk resume screening for recruitment agencies.

ACKNOWLEDGEMENT

The authors would like to thank the Department of Electronics & Communication Engineering, Lakshmi Narain College of Technology & Science, Bhopal, for their support and guidance in the completion of this research work.

REFERENCES

[1] T. Brown et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems (NeurIPS), 2020.

[2] Groq Inc., "Groq API Documentation," [Online]. Available:https://console.groq.com/docs. [Accessed: 2024].

[3] Meta AI, "LLaMA 3: Open Foundation and Fine-Tuned Chat Models," Meta AI Research, 2024.

[4] OpenJS Foundation, "Node.js Documentation," [Online]. Available:https://nodejs.org/en/docs. [Accessed: 2024].

[5] R. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proceedings of NAACL-HLT, 2019.

[6] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[7] Resend, "Resend API Documentation," [Online]. Available: https://resend.com/docs. [Accessed: 2025].