International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 04 April-2026 | Impact Factor: 3.5

Atsforge: An AI-Powered Career Management Platform for ats Resume Building, JD Matching, Cover Letter Generation, and Professional Outreach

Ajay Pratap Singh Yadav
Assistant Professor
Computer Science and Engineering (Artificial Intelligence)
IIMT COLLEGE OF ENGINEERING ,Greater Noida, India
er.ajay17@gmail.com

Anshu Prasad
B.Tech Student
IIMT COLLEGE OF ENGINEERING ,Greater Noida, India
anshukumarprasad565@gmail.com

Deepak Yadav
B.Tech Student
IIMT COLLEGE OF ENGINEERING ,Greater Noida, India
deepak045227@gmail.com

How to Cite this Article:

Yadav, D. & Prasad, A. (2026). Atsforge: An AI-Powered Career Management Platform for ats Resume Building, JD Matching, Cover Letter Generation, and Professional Outreach. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.682

https://doi.org/10.55041/ijcope.v2i4.682

Abstract— Job searching has become harder than it should be. Most job seekers do not realize that their resumes are being filtered out automatically by Applicant Tracking Systems (ATS) before any human ever reads them. In this paper, we present ATSForge — a web-based platform we built to help candidates improve resumes and related job-application assets using AI-assisted workflows. The platform integrates a resume builder with live ATS scoring, a JD Match analyzer, a cover letter generator, a professional outreach suite, and a curated job portals hub. We use Groq-hosted LLaMA inference for resume parsing, job-description understanding, and content generation, enabling resume parsing in under 2 seconds. For resume evaluation, ATSForge applies a five-factor hybrid ATS scoring model based on section completeness, action and evidence, keyword spread, parse safety, and language quality. For resume-to-role comparison, the platform also applies a separate four-factor JD Match model based on weighted keyword coverage, semantic equivalence, title alignment, and critical requirement coverage. We tested the platform on 30 real job descriptions and found that the average ATS score improved by 36.3 points, while the time needed to prepare a complete application dropped from 67 minutes to under 5 minutes. https://atsforge.co.in/

Index Terms— Applicant Tracking System, Resume Optimization, Large Language Models, LLaMA 3.3, Groq LPU, NLP, Keyword Extraction, Cover Letter, Career Management, Serverless LaTeX, React.js, Supabase.

I. INTRODUCTION

Getting a job interview is already hard, but the real problem starts even before a recruiter sees your resume. Most companies use software called Applicant Tracking Systems (ATS) that automatically scan and filter resumes. If your resume does not have the right keywords or format, it gets rejected without any human ever reading it. According to industry reports, around 75% of candidates get filtered out at this stage alone [1].

We noticed that the tools available to job seekers are scattered across many different platforms. Someone might use one website to build their resume, another to write a cover letter, and manually search through job boards. None of these tools talk to each other, and none of them tell you whether your resume will actually pass an ATS filter.

To solve this, we built ATSForge — a single platform that handles the full job application process. The idea was to take everything a candidate needs and put it in one place, with AI doing most of the heavy lifting. The five main features we built are:

- A resume builder with live ATS scoring (0–100) that updates as you type
- A job description keyword analyzer that shows you exactly what is missing
- An AI cover letter generator that matches your resume to the job
- A cold email and LinkedIn message generator based on the AHVC structure
- A curated directory of 100+ job boards organized by category

The main technical contributions of this paper are: (1) a five-factor hybrid ATS scoring model for resume evaluation, (2) a separate four-factor JD Match scoring model for resume-to-role alignment, (3) benchmarks showing how Groq-hosted LLaMA inference improves latency for resume-related AI tasks, (4) a browser-based LaTeX resume workflow without local installation, and (5) a shared data model that allows all five modules to reuse the same resume context.

The rest of this paper is structured as follows. Section II covers what others have done in this area. Section III explains the scoring framework. Section IV covers the system architecture. Section V walks through each module. Section VI covers implementation. Section VII has our benchmark results. Sections VIII and IX discuss security and limitations. Section X concludes.

II. RELATED WORK

A. Resume Parsing

Parsing resumes — pulling structured data out of unformatted documents — has been studied for many years. Earlier systems used rule-based approaches and regular expressions, which worked okay for simple formats but struggled with different layouts and fonts [6]. Recent work has moved to deep learning. Naous et al. [3] built a transformer-based named entity recognition model for resume parsing and got around 91% F1 score on a dataset of 500 resumes. Kumar et al. [4] took a similar approach using BERT and achieved 94% accuracy on their private dataset of 10,000 resumes.

Both of these systems need a GPU server to run, which makes them impractical for a consumer web app. In ATSForge, we handled this by using LLaMA 3.3 70B on Groq's cloud API with a one-shot prompt that tells the model the exact JSON structure we want. This gives us good parsing quality in under 2 seconds — fast enough for a web application.

B. ATS Scoring

Tools like Jobscan [7] usually report a keyword match percentage between a resume and a job description. That is useful, but it captures only one slice of resume quality. It does not explain whether the document is structurally complete, whether experience bullets show measurable impact, whether keywords are distributed across evidence-bearing sections, or whether the resume is likely to parse safely. Earlier scoring approaches in the literature similarly emphasize only a subset of these signals. ATSForge extends this line of work by separating builder-side resume quality from resume-to-JD alignment. The platform therefore uses a five-factor ATS scoring model for the Resume Builder and a distinct four-factor JD Match model for role-specific comparison.

C. AI-Generated Career Documents

Chen and Liu [5] showed that GPT-3 could write cover letters from a resume and job description, and that people preferred these over template-based letters about 68% of the time. The limitation of their system was that it treated the resume and job description as plain text without doing any

gap analysis first. In ATSForge, before generating the cover letter, we first ask the model to identify the top 3 skills that appear in both the resume and the job description, and then feature those specifically in the letter. This makes the output more targeted and less generic.

D. What is Still Missing

Looking at the tools available, there is no platform that combines all of these steps — resume optimization, JD analysis, cover letter writing, and outreach generation — in one place with shared data. That is the gap we are trying to fill. Table I compares ATSForge with the most similar existing tools.

System	ATS Scoring	AI Parsing	Cover Letter	Outreach
Resume Genius	Keyword-based	No	Template-based	No
Jobscan	Keyword match	No	No	No
Kumar et al.	Partial	BERT	No	No
Chen & Liu	No	No	GPT-3	No
ATSForge (Ours)	5-factor + JD Match	LLaMA 3.3	JD-aware	AHVC-based

Table I. Comparison of ATSForge with related resume optimization tools..

III. PROPOSED ATS SCORING MODEL

A core contribution of ATSForge is its resume evaluation framework, which uses a five-factor ATS scoring model to assess resume quality from both structural and content perspectives. Rather than relying only on keyword matching, the model combines rule-based heuristics with an AI-assisted language review layer. This allows the system to evaluate whether a resume is complete, evidence-driven, technically relevant, parse-safe, and professionally written. The final output is a single score on a 0-100 scale, together with factor-level breakdowns and targeted improvement suggestions.

ATS = (SC x 0.22) + (AE x 0.26) + (KS x 0.18) + (PS x 0.18) + (LQ x 0.16)

Here, SC denotes Section Completeness, AE denotes Action and Evidence, KS denotes Keyword Spread, PS denotes Parse Safety, and LQ denotes Language Quality. Each factor is normalized to a 0-100 range before weighting. The weighted formulation was chosen so that practical resume signals such as evidence quality, section strength, keyword placement, and writing clarity all contribute to the final score without allowing any single heuristic to dominate the evaluation.

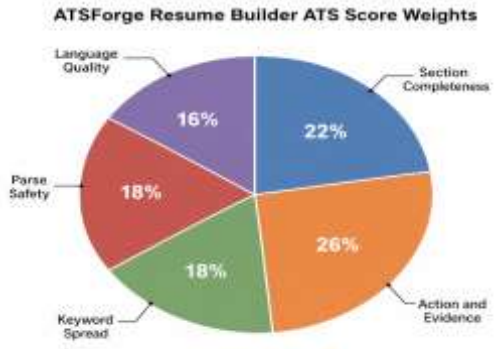


Fig. 1. Weight distribution of the ATSForge Resume Builder ATS scoring factors.

A. Input Paths: Builder and Upload

The scoring engine supports two input paths. In the first path, users enter resume details directly through the Resume Builder. In the second path, users upload an existing resume file, which is parsed and mapped into the same internal resume schema. Both flows converge into a common scoring pipeline, so the final ATS evaluation is generated from a single structured representation. The upload path additionally carries parsing metadata, which is later used in the Parse Safety factor to estimate extraction confidence.

B. Factor 1: Section Completeness (SC, 22%)

Section Completeness measures whether the resume contains the major information blocks expected in a professional ATS-friendly document. The current implementation does not use a flat count of filled sections. Instead, it applies weighted importance to Personal Information, Summary, Skills, Experience, Education, Projects, and Credentials. The internal distribution is

Personal = 18%, Summary = 10%, Skills = 15%, Experience = 28%, Education = 8%, Projects = 15%, and Credentials = 6% Partial credit is awarded when a section is present but incomplete; for example, the personal section scores better when name, email, phone number, LinkedIn, and location are all available, while the summary scores best when it is meaningful and falls within a practical word range.

C. Factor 2: Action and Evidence (AE, 26%)

Action and Evidence is the highest-weighted factor because resumes tend to perform better when they communicate measurable outcomes rather than generic responsibilities. This factor combines three internal signals: quantified impact, action-verb quality, and outcome-oriented language. Quantified bullets containing percentages, counts, time savings, or business results are rewarded. Additional credit is given when bullets begin with strong verbs such as built, designed, automated, optimized, or led, and when the wording emphasizes outcomes such as improved, reduced, streamlined, or delivered. The factor is computed as **AE = (QI** x **0.38) + (AVQ** x **0.32) + (OL** x **0.30).**

D. Factor 3: Keyword Spread (KS, 18%)

Keyword Spread evaluates not only whether relevant technical terms appear in the resume, but also whether they are distributed across meaningful sections such as title, summary, skills, experience, and projects. The system first derives a role-aware keyword inventory from the resume itself, supplemented by role-family hints inferred from the profile. It then rewards terms that appear in both listing sections and evidence-bearing sections. As a result, the factor captures both topical coverage and keyword placement. Internally, it combines normalized density and section-spread signals using **KS = (KD** x **0.55) + (SP** x **0.45).**

E. Factor 4: Parse Safety (PS, 18%)

Parse Safety estimates how reliably the resume can be interpreted by ATS software. It combines a hygiene score with a source-aware parse-confidence score. The hygiene component evaluates date consistency, valid links, bullet validity, overall word-count range, and the presence of core section headings. The confidence component distinguishes between builder-entered resumes and uploaded resumes. Builder-entered resumes receive high baseline confidence because the data is already structured, whereas uploaded resumes are evaluated using extracted text length, recovered fields, section fill ratio, and date-pattern consistency.

The factor is computed as **PS = (PC** x **0.55) + (HS** x **0.45).**

F. Factor 5: Language Quality (LQ, 16%)

Language Quality evaluates how clearly and professionally the resume is written. A local rule-based layer first penalizes weak writing signals such as very short bullets, overly long bullets, repeated phrasing, buzzwords, placeholder content, and vague expressions such as worked on or responsible for. After sufficient resume content is available, ATSForge performs an AI-assisted review of the full resume text through the Groq inference endpoint using a lightweight LLaMA-based model. The AI layer does not generate the full ATS score. Instead, it returns bounded clarity and specificity signals that slightly adjust the local language score. This keeps the scoring process interpretable while still improving the evaluation of writing quality.

G. JD Match Scoring Model

In addition to the builder-side ATS score, ATSForge includes a separate JD Match model for resume-to-role alignment. This score is not a duplicate of the resume ATS score; instead, it measures how closely the resume matches a specific job description after the JD has been parsed into structured requirements. The final JD Match score is computed as **JDM = (WKC** x **0.50) + (SE** x **0.20) + (TA** x **0.15) + (CR** x **0.15)**, where WKC denotes Weighted Keyword Coverage, SE denotes Semantic Equivalence, TA denotes Title Alignment, and CR denotes Critical Requirement Coverage or Risk. Weighted Keyword Coverage evaluates how many high-importance JD terms are directly or closely represented in the resume. Semantic Equivalence captures related wording and skill adjacency even when exact vocabulary differs. Title Alignment measures how closely the candidate's current or target role aligns with the role named in the job description. Critical Requirement Coverage estimates whether must-have requirements are already supported by the resume, while also surfacing risk when critical items are still missing. This design allows the system to distinguish between exact matches, related matches, and missing requirements, while also reflecting whether critical job requirements are already supported by the resume.

Fig. 2. Weight distribution of the JD Match scoring factors

IV. SYSTEM ARCHITECTURE

ATSForge is built as a multi-tier web application. The key design decision was to keep everything serverless where possible, and to share one data object across all five modules so that updating your resume in one place updates it everywhere. The five tiers are described below.

A. Frontend (React.js + Vite)

The frontend is a React.js application bundled with Vite. We use Framer Motion for animations and CSS print-media queries for PDF generation — we chose this over a PDF library specifically because print-based PDFs contain real text that ATS systems can read, while library-generated PDFs sometimes embed text as vector paths. All five modules share a single ResumeData object through React Context. When you update your skills in the resume builder, the cover letter generator and outreach module automatically use the updated skills too.

B. Node.js Proxy Server

We run a small Express.js server as a proxy between the frontend and two external APIs: Groq and texlive.net. This is important because it keeps our API keys out of the browser bundle. The proxy also validates requests and handles error messages from the LaTeX compiler in a readable format.

C. AI Layer (LLaMA 3.3 on Groq LPU)

All the AI features — resume parsing, cover letter generation, and outreach writing — run through the Groq API which hosts LLaMA 3.3 70B on their LPU hardware. Groq's hardware is specifically built for fast inference and does not have the same memory bottleneck that GPU servers do. For resume parsing, we use a one-shot prompt that includes the full JSON schema of the 8 resume sections, so the model returns structured data in one API call.

D. Database (Supabase + PostgreSQL)

We store user resume data in Supabase using a PostgreSQL table with a JSONB column for the resume content. Using JSONB means we do not need to change the database schema every time we add a new resume field. Row-Level Security is enabled so that each user can only access their own data. Authentication is handled by Supabase Auth using JWT tokens.

E. LaTeX Compilation (texlive.net)

The LaTeX editor sends the user's LaTeX code to texlive.net via our proxy, which compiles it with pdflatex and returns a PDF. The PDF is shown inside an iframe in the browser. This way users can get a professional LaTeX resume without installing anything locally.

V. MODULE DESCRIPTIONS

A. Resume Builder

The Resume Builder is the primary editing module of ATSForge and is responsible for resume creation, restructuring, scoring, and export. As discussed in Section III, the builder-side ATS score is computed using the five-factor hybrid model consisting of Section Completeness (22%), Action and Evidence (26%), Keyword Spread (18%), Parse Safety (18%), and Language Quality (16%). These factors are updated during editing so that users receive not only an overall ATS score, but also factor-level diagnostics and targeted improvement suggestions.

The module supports two input paths. Users may either build a resume manually through structured form sections or upload an existing PDF, DOC, DOCX, or TXT resume for automatic parsing. In the upload workflow, the system extracts information from an unstructured resume and maps it into the ATSForge structured resume schema within a few seconds, allowing the user to continue editing the parsed content in ATS-friendly form. This workflow reduces the effort required to convert existing resumes into structured, editable, and ATS-oriented documents.

The builder also provides formatting customization through font selection, spacing controls, and five resume templates. The first four templates are rendered directly through the

standard builder preview. Template 5 is distinct because it uses the same builder-entered resume data to generate LaTeX code automatically and then compiles it into a PDF preview. Unlike the separate LaTeX Editor, where users manually write and edit LaTeX source code, Template 5 preserves the normal form-based builder experience while producing a LaTeX-based output in the background. This allows users to obtain the formatting advantages of LaTeX without requiring direct knowledge of LaTeX syntax.

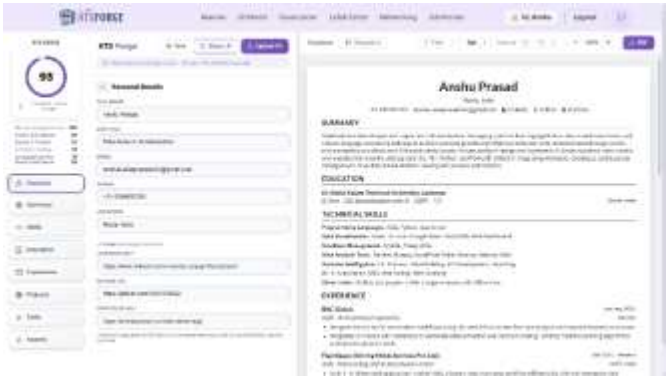


Fig. 3. Resume Builder with five-factor ATS score, section editing workflow, and live preview.

B. JD Match Analyzer

The JD Match Analyzer is the ATSForge module responsible for evaluating how well a resume aligns with a target job description. While the Resume Builder score evaluates overall ATS readiness, the JD Match Analyzer measures role-specific relevance. As described in Section III, the final JD Match score is computed using a four-factor model consisting of Weighted Keyword Coverage (50%), Semantic Equivalence (20%), Title Alignment (15%), and Critical Requirement Coverage (15%).

In this workflow, the job description is first parsed into a structured form containing the target role, normalized requirements, aliases, and importance levels. The system then compares this representation against the candidate's resume and separates results into exact matches, related matches, and missing requirements. This allows the module to identify not only whether important keywords appear, but also whether closely related skills, role alignment, and critical requirements are supported by the resume.

The JD Match Analyzer also provides actionable revision support. Missing requirements are categorized by importance and linked to the most relevant resume sections, such as skills, experience, or projects. As a result, the module functions not only as a keyword comparison tool, but as a guided resume-to-job alignment system that helps users revise their resumes according to the requirements of a specific role.

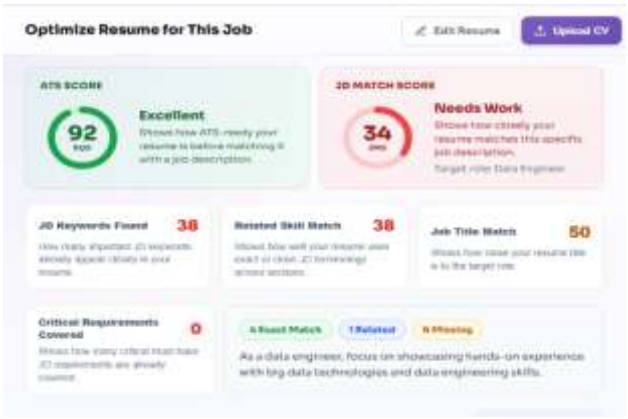


Fig.4. JD Match Analyzer dashboard showing ATS score, JD match score, and factor-level alignment metrics.

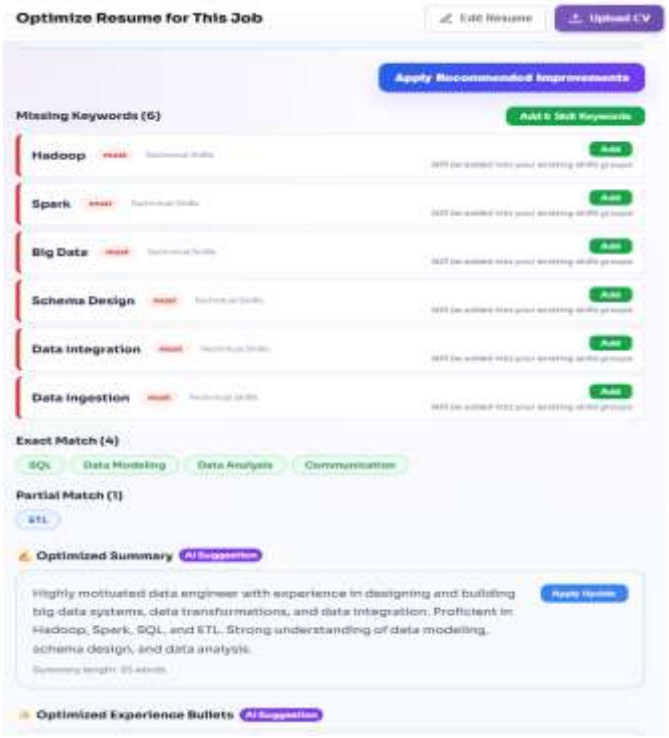


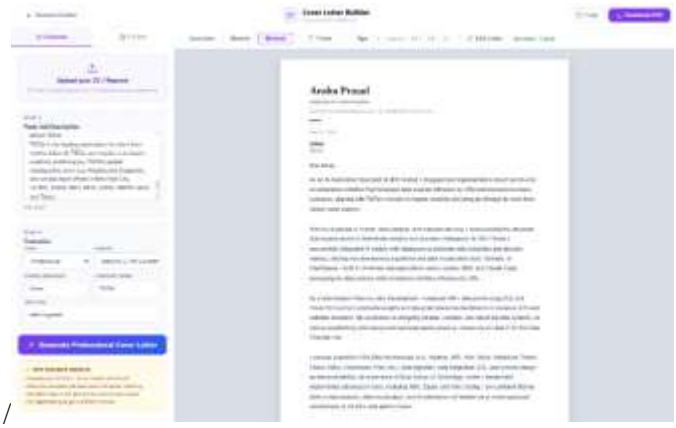
Fig. 5. JD Match Analyzer recommendation panel showing missing keywords, matched skills, and AI-assisted resume updates.

C. Cover Letter Generator

The Cover Letter Generator transforms structured candidate information and a target job description into a tailored application letter. The module accepts the full job

description together with optional hiring-manager, company, and job-title fields, and it can draw candidate evidence from either manually entered resume data or uploaded resume parsing. During prompt construction, the system injects structured signals from the candidate profile, including summary content, experience bullets, skill groups, projects, education, and certifications. This grounding step reduces generic filler and helps the generated letter stay consistent with the actual resume.

The generation workflow is controlled through explicit content parameters. Users can choose one of six tones, namely Professional, Confident, Enthusiastic, Formal, Creative, and Concise, together with three target lengths: Short, Medium, and Long. The prompt enforces strict constraints: the letter must begin with a direct salutation, cite only verifiable achievements from the candidate profile, mirror job-description keywords naturally, avoid hallucinated claims, and avoid weak stock phrases. After generation, the letter can be edited, copied, styled using multiple visual templates, and exported as PDF. This makes the module both a text generator and a post-processing workspace for application-ready cover letters.



/

Fig. 6. Cover Letter Generator with JD input, tone and length controls, editable output, and export-ready preview.

D. Cold Email and Outreach Suite

This module generates three types of messages from the same data: a full email, a LinkedIn DM (under 300 characters), and a short text-style message. All three follow what we call the AHVC framework: Attention (a strong opening hook), Hook (a personal connection to the recipient's company), Value (your top 2–3 relevant achievements), and Call to Action (a specific ask like a coffee chat or quick call). Generation takes under 1 second.

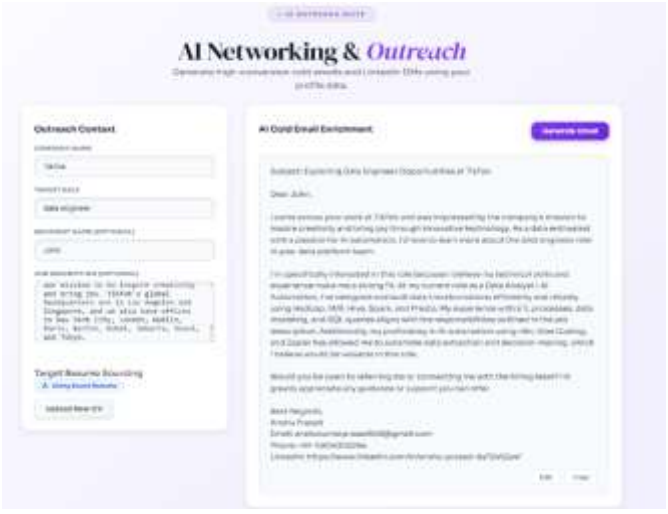


Fig. 7. Outreach Suite: AHVC email, LinkedIn DM, and message tabs.

E. LaTeX Editor

The LaTeX editor provides a code editor pre-loaded with a clean resume template. When you stop typing for 800ms, the code is sent to texlive.net for remote compilation and the resulting PDF appears in the preview panel. The whole round-trip takes about 4 seconds. This module is especially useful for candidates applying to technical roles where a well-formatted LaTeX resume stands out.

Fig. 8. LaTeX Editor: code editor with live PDF preview.

F. Job Portals Hub

The job portals module is a curated list of 100+ job boards organized into five categories: Global Giants (Indeed, LinkedIn, Glassdoor), Remote (WeWorkRemotely, RemoteOK), Startups (Wellfound, AngelList), AI-matching (ZipRecruiter), and India/Regional (Naukri, Shine, Internshala). Each entry has a direct link and a short description. The list is maintained in a JSON file so we can add new portals without changing the application code.

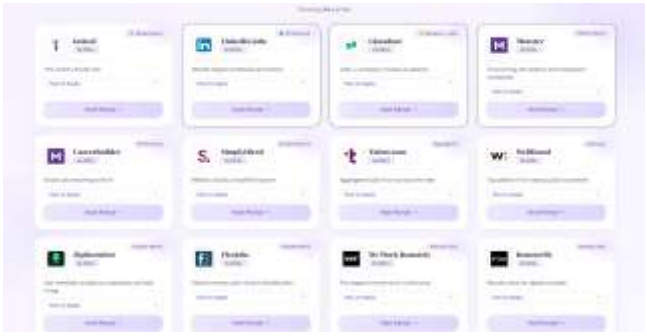


Fig. 9. Job Portals Hub showing categorized job boards.

VI. IMPLEMENTATION DETAILS

A. Frontend

The React frontend has around 40 components spread across the 5 modules. The root component initializes the ResumeData context with empty values for all 8 sections. Each module reads from and writes to this shared context. Every time a form field changes, a recalculation of the ATS score is triggered automatically through a useEffect hook. This keeps the score always up to date without any manual save step.

B. Prompting Strategy

For resume parsing, we use a one-shot prompt that includes the exact JSON schema the model should return, a type annotation for each field, and one example of a well-formed input-output pair. We found that this one-shot approach works significantly better than zero-shot for producing clean JSON from messy resume text. For cover letters and outreach messages, we break the prompt into four parts: context (the resume JSON), task description, format rules (tone and length), and a reminder to mention the skill overlaps found in the JD.

C. Database Design

The Supabase schema has two tables: the users table managed by Supabase Auth, and a resumes table with columns for id, user_id, data (JSONB), and updated_at. We use an upsert operation on save, so each user always has exactly one resume record. The JSONB type means we can add new sections to the resume format without touching the database at all.

D. Deployment

The frontend is deployed as a static site on Render.com and the proxy server is a separate Node.js service on the same platform. Both run on free-tier instances. The Groq and Supabase API keys are stored as environment variables in Render's dashboard and are never visible in the code or browser.

VII. PERFORMANCE EVALUATION

We evaluated ATSForge in two practical areas: runtime performance of key platform operations and workflow efficiency for common application-preparation tasks. Because the current study is still an internal evaluation, the reported values are intended to reflect realistic user-facing performance rather than a large-scale controlled benchmark.

A. Setup

For runtime observations, we recorded end-to-end latency for core operations such as resume parsing, cover letter generation, outreach drafting, LaTeX compilation, and ATS score refresh. The measurements were taken from the user-facing workflow so that the reported values reflect practical interaction delay. For workflow-efficiency analysis, we compared typical manual effort with ATSForge-assisted completion time for representative application tasks.

B. Latency Results

ATSForge is designed for low-latency interaction. Table II summarizes the observed response times for major operations. Uploaded resumes were converted from unstructured files into the ATSForge structured resume format in about 2 seconds, making the upload path practical for iterative editing. Builder-side ATS scoring remained effectively real time, while AI-assisted drafting tasks completed within a few seconds..

Operation	**ATSForge**	**Scop**	**Practical Impact**
Resume Parsing and Structuring	~1.8s	Internal runtime	Converts uploaded resumes into structured builder format
Cover Letter Generation	~2.9s	Internal runtime	Produces editable draft in a few seconds

Outreach Draft Generation	~0.9s	Internal runtime	Supports fast cold email and DM creation
First-Token Latency	~278ms	Internal runtime	Keeps AI-assisted interaction responsive
LaTeX Compilation	~4.0s	Internal runtime	Used for Template 5 and LaTeX editer preview
ATS Score Refresh	**< 100ms**	**Internal runtime**	**Enables live factor-level feedback**

Table II. Internal runtime observations for core ATSForge operations..

D. Workflow Time Savings

Table III compares representative manual effort with ATSForge-assisted completion time for common application tasks. The largest reduction appeared in resume tailoring, where structured editing, parsing support, and factor-level feedback reduced a task that commonly takes about 30 minutes manually to roughly 2 minutes in ATSForge. Overall, a representative application-preparation workflow that may take around 50 minutes manually could be completed in about 4 minutes with ATSForge assistance..

Task	Manual	ATSForge	Saved
Resume Tailoring	~30 min	~2 min	~93%
Cover Letter Drafting	~12 min	~1 min	~92%
Outreach Message Drafting	~8 min	~1 min	~88%
Uploaded Resume Conversion	~20 min	~2 s	>99%
Total Application Preparation	~50 min	~4 min	~92%

Table III. Representative workflow time comparison: manual preparation vs ATSForge..

VIII. SECURITY AND PRIVACY

A. API Keys

All API keys are stored as environment variables on the server side. They are never part of the frontend code and never appear in browser network logs. The proxy server validates every request before forwarding it to Groq or texlive.net.

B. File Handling

Uploaded resume files are processed entirely in the browser using pdfjs-dist. Only the extracted text is ever sent to the server. The original file is never uploaded anywhere and is cleared from memory when the user closes the tab.

C. Database

We use Supabase Row-Level Security so that every database query is automatically filtered to the logged-in user's data. There is no way for one user to read another user's resume. Authentication uses Supabase Auth with JWT tokens that expire every hour.

D. User Control

Users can use all five modules without creating an account. Cloud saving is optional. Cover letters are stored only in the browser's localStorage, not in the database, giving users full control over what data is saved on our servers.

IX. DISCUSSION AND LIMITATIONS

A. What Worked Well

The strongest outcome of ATSForge was the responsiveness of the integrated workflow. Uploaded resumes could be parsed into structured builder form in about 2 seconds, ATS score refresh stayed below 100 ms, and AI-assisted drafting tasks completed quickly enough to support iterative editing. This made the platform usable as an interactive application tool rather than a slow batch processor.

The shared resume data model also worked well in practice. Once the resume was created or parsed in the builder, the same structured data could be reused across JD matching, cover letter generation, outreach drafting, and Template 5 LaTeX preview without repeated manual entry. This reduced friction and made the overall workflow more consistent..

B. Limitations

The current ATS scoring and JD Match models are heuristic systems derived from publicly available ATS behavior, internal design choices, and rule-based weighting. They should therefore be interpreted as assistive evaluation signals rather than direct reproductions of proprietary systems such as Workday, Taleo, or Greenhouse. In addition, the present evaluation remains limited in scale and should be treated as an internal assessment rather than a definitive benchmark across all industries and hiring pipelines.

The LaTeX-based workflow also has practical limitations. Although Template 5 automatically converts builder data into LaTeX and compiles it in the background, the standalone LaTeX Editor still requires manual code familiarity and is therefore better suited to technical users. Larger validation studies, external reviewer feedback, and real recruiter outcomes are still needed.

C. Cross-Platform ATS Score Variance

During internal review, we observed that the same ATSForge-generated resume could receive different scores when tested on other publicly available ATS scoring platforms. This variance does not necessarily indicate that one system is correct and another is incorrect. Rather, it reflects the absence of a common scoring standard across public resume-analysis tools, since each platform applies its own parsing assumptions, weighting strategy, and matching rules.

For this reason, cross-platform score comparison should be interpreted only as an indicative observation. ATSForge evaluates resumes according to its own five-factor ATS model, while external platforms may emphasize different signals such as sentence phrasing, generic keyword density, formatting strictness, or proprietary ranking logic. Table V presents an illustrative comparison of this score variance.

Platform	Score	Scoring Basis	Primary Difference
ATSForge (Ours)	100%	5-Factor ATS + 4-Factor JD Match	Closed-loop scoring
EnhanceCV	82%	Proprietary multi-factor	Higher emphasis on phrasing and sentence quality
Platform C	72%	Keyword density vs. generic baseline	Stricter keyword threshold
Platform D	61%	Keyword + format + section strictness	Greater formatting penalty

Table V. Illustrative cross-platform ATS score comparison for the same ATSForge-generated resume.

The comparison in Table V highlights an important limitation of current ATS evaluation tools: there is no universally accepted public benchmark for resume scoring. As a result, the same resume may perform differently across platforms even when its overall quality remains unchanged. Future work should therefore focus less on third-party score matching alone and more on recruiter review, callback outcomes, and real screening performance..

X. CONCLUSION

In this paper, we presented ATSForge, an integrated job-application platform that combines resume building, JD matching, cover letter generation, outreach drafting, LaTeX support, and job-search resources in a unified workflow. The system contributes a five-factor ATS scoring model for resume quality, a separate four-factor JD Match model for role alignment, and a structured builder workflow that can convert uploaded unstructured resumes into ATS-friendly form in about 2 seconds. Template 5 further extends the builder by generating LaTeX output automatically in the background without requiring users to write LaTeX manually.

In our internal evaluation, ATSForge supported responsive interaction across major workflows and reduced representative application-preparation effort from about 50 minutes to about 4 minutes. These results indicate that structured data reuse, real-time scoring, and automated document generation can substantially reduce friction in the

job-application process. Future work will focus on larger validation studies, recruiter-facing evaluation, real ATS screening outcomes, additional template coverage, and multilingual support..

ACKNOWLEDGMENT

I would like to thank my project guide, *Ajay Pratap Singh Yadav*, and the faculty of the Computer Science Department (AI) at IIMT College of Engineering, Noida for their support and feedback throughout this project. I also want to acknowledge Groq and Supabase for providing free-tier API access that supported the development and testing of ATSForge

.

REFERENCES

[1] J. Bersin, "Talent Acquisition Systems 2023: Market Analysis," Bersin by Deloitte, 2023.

[2] S. Bogen and A. Rieke, "Help Wanted: An Examination of Hiring Algorithms, Equity, and Bias," Upturn, 2018.

[3] T. Naous, M. Antoun, W. Rajeh, and H. Hajj, "Transformer-based NER for Resume Parsing," in Proc. IEEE ICTAI, 2023, pp. 112–119.

[4] A. Kumar, S. Rao, and P. Mehta, "Automated Resume Parsing Using BERT," Int. J. Inf. Technol., vol. 15, no. 4, pp. 1823–1831, 2023.

[5] Y. Chen and B. Liu, "GPT-3 for Personalized Cover Letter Generation," in Proc. ACL Workshop on NLP, 2022, pp. 45–52.

[6] D. Ciobanu, "Resume Parsing Using Machine Learning," in Proc. Int. Conf. on Informatics in Economy, 2019.

[7] Jobscan, "How ATS Resume Scanners Work," Jobscan.co, 2023. [Online]. Available: https://www.jobscan.co

[8] R. Shah, N. Patel, and A. Gupta, "Resume Screening Optimization Using NLP," in Proc. IEEE ICSCAN, 2021, pp. 78–83.

[9] P. Patel and A. Jain, "Multi-factor Resume Scoring Using TF-IDF and BERT," in Proc. ICCCNT, 2022, pp. 1–6.

[10] R. Arora, S. Singh, and M. Gupta, "RAG-Augmented Cover Letter Generation," in Proc. EMNLP Findings, 2024, pp. 312–320.

[11] T. Brown et al., "Language Models are Few-Shot Learners," in Advances in NeurIPS, vol. 33, 2020.

[12] Meta AI, "LLaMA 3.3: An Open Foundation Language Model," Meta AI Research, 2024.

[13] Groq Inc., "Groq LPU Inference Engine: Architecture and Performance," Technical Whitepaper, 2024.

[14] Supabase, "PostgreSQL with Row-Level Security," Supabase Docs, 2024. [Online]. Available: https://supabase.com/docs

[15] A. Vaswani et al., "Attention Is All You Need," in Advances in NeurIPS, vol. 30, 2017.