AI-Powered Resume Builder Using Google Gemini Flash Model and Node-React Stack

Abhishek Mathur[1], Anuradha Misra[2]
[1,2]*Amity School of Engineering and Technology, Amity University Uttar Pradesh, Lucknow, India*

***Abstract*—The resume builder project which is an AI-Powered Resume Builder presents an AI-driven platform to streamline professional resume creation by combining modern web technologies with advanced language models. Leveraging a React/Tailwind CSS frontend and a Node.js/Express backend with a PostgreSQL database, the system uses Google's Gemini Flash 2.0 API to generate resume content from minimal user input. Generated resumes are immediately displayed in a live editor and evaluated for ATS (Applicant Tracking System) compliance. A companion Telegram bot allows users to retrieve resumes via chat. This paper details the system's architecture, methodology, and implementation. We demonstrate how AI assistance can greatly reduce the time and effort required to craft high-quality, ATS-optimized resumes. A qualitative evaluation indicates improvements in user efficiency and resume quality, aligning with prior findings that algorithmic writing assistance increases hiring success. The paper concludes with a discussion comparing the AI-aided workflow to traditional resume writing and proposes future enhancements such as multilingual support, LinkedIn integration, and PDF export.**

***Index Terms*—AI resume builder, Google Gemini, React, Node.js, PostgreSQL, ATS optimization, Telegram bot**

I. INTRODUCTION

Writing an effective resume remains a major challenge for job seekers. Traditional resume creation is manual and error-prone, often failing to highlight relevant skills or to meet the stringent requirements of applicant tracking systems (ATS)[1][2]. Modern employment processes frequently use ATS software to scan and score resumes, which means unoptimized resumes can be filtered out regardless of candidate merit[3][4]. Consequently, many well-qualified applicants never reach recruiters simply due to poor resume formatting or missing keywords[3][4]. At the same time, recruiters face large volumes of applications, making manual screening inefficient and error-prone[5].

Recent studies have shown that algorithmic writing assistance can significantly improve resume quality and hiring outcomes. For example, van Inwegen et al. (MIT) found that resumes improved with AI editing received 7.8% more job offers and were more likely to result in hiring[24]. This underscores that employers care about writing quality, and that AI tools can level the field for non-native speakers or less experienced writers[2]. These insights motivate the proposed system: an AI-powered resume builder that automatically generates and optimizes content, helping users craft tailored, ATS-friendly resumes in seconds. Several researches have shown that automated systems can reduce the effort needed to produce professional resumes and improve ATS compliance[6][7].

The Resume Generator and ATS Analyzer enables users to input basic information (name, email, target job title, etc.) and then relies on Google Gemini Flash 2.0 a state-of-the-art language model to generate detailed sections (summary, skills, experience)[14][15]. An intuitive React/Tailwind live editor lets users immediately preview and refine the resume. An integrated ATS checker provides a compliance score and keyword recommendations. The complete system includes an optional Telegram bot to deliver completed resumes via chat. The user needs to start a conversation with the bot and then the bot grabs the chat ID which it uses to deliver the generated resumes. In summary, our resume system is capable of automating resume drafting while preserving user control, aiming to empower job seekers and facilitate recruiters' workflows by producing higher-quality, more discoverable resumes.

II. RELATED WORK

Several online services offer resume templates or basic keyword recommendations, but few integrate advanced AI generation. Traditional tools like ResumeGenius or LinkedIn Resume Builder rely on fixed templates and manual input. Recently, a new class of AI-powered resume builders has emerged. For example, CleverCV[3] and similar projects leverage generative AI (e.g. GPT variants) to automatically compose resumes. These systems report improvements in compliance with ATS filtering: tailored, AI-generated content tends to match job descriptions more closely than user-written text[6].

Academic and industry research also underscores the value of AI in recruitment. The work by van Inwegen et al.[2] highlights that small improvements in writing quality can significantly affect hiring rates. Other studies (e.g. IRJMETS2025) describe systems combining React frontends with AI backends and ATS scoring, showing higher ATS scores and user satisfaction when resumes are AI-enhanced[8][6]. In the broader context, AI in recruitment spans chatbot screeners, skill assessments, and automated resume screening. Our system builds on these insights by explicitly combining a modern tech stack (React + Node.js + PostgreSQL) with Google's newest Gemini model for content generation. Unlike prior work that used GPT or older models, this system uses Google Gemini Flash 2.0 for potentially better contextual understanding[13][15].

III. METHODOLOGY

A. System Architecture

The Kenshi Resumes platform follows a client-server architecture with four key components: a web-based frontend, a backend API server which also houses two separate AI modules for ATS Scoring and providing further recommendations to the user for improving the current resume, a database using PostgreSQL, and auxiliary services (Google Gemini API and Telegram bot). Figure 1 depicts the high-level architecture.

The frontend is built in React (initialized via Vite) with Tailwind CSS for styling and Shadcn library for UI components which are resuable. React provides a dynamic single-page interface, including a live resume editor and form inputs. As users edit fields or click "Generate," the UI issues HTTP requests to the backend. Tailwind ensures a responsive design across devices. The frontend is statically hosted and deployed on Vercel, which offers automatic builds, serverless functions and global CDN distribution.

The backend runs on Node.js with the Express framework. Node.js provides a scalable, event-driven runtime for JavaScript on the server, and Express supplies routing and middleware support. The backend exposes a RESTful API (prefixed with /api) to the frontend. Key routes include user resume CRUD (create, read, update, delete), AI generation triggers, ATS checking, and Telegram bot interactions. For example, POST /api/user-resumes creates a new resume entry, PUT /api/user-resumes/:id updates sections, and POST /api/user-resumes/ats/:id submits a file for ATS scoring.

User data and generated content are persisted in a PostgreSQL database, a robust open-source relational system[16]. The schema includes tables such as user resumes, experience, education, skills, and telegramusers[16]. Each resume is linked to a user email, and related sections are stored in joined tables. Postgres ensures reliability and supports our future scalability needs.

The AI functionality relies on Google Gemini Flash 2.0 via its API[14]. Gemini is a cutting-edge multimodal language model with advanced reasoning capabilities[9][18]. In our resume system, the backend calls Gemini's text generation endpoint (using the user's job title and inputs) to produce draft content for each resume section. It also uses a custom ATS recommendations prompt which uses the FlashRecommnedations.js module to identify missing keywords.

Finally, a Telegram bot (implemented in bot.js) allows users to receive their final resume via chat[19]. When the user requests it, the backend retrieves the PDF/buffer from the database and sends it through Telegram's API to the user's account. The bot adds convenience by enabling mobile access to the resume. Overall, the whole system's architecture integrates modern web services with AI and messaging seamlessly[1].

The workflow of AI Resume system proceeds in several stages, as illustrated in Figure 2. For user authentication, our system uses the Clerk library which encrypts the system's sensitive information using the bcrypt hashing algorithm. It also uses digitally signed JWT (JSON Web Tokens) which uses

RS256 (RS + SHA-256) which is an asymmetric algorithm thereby ensuring that the token generated is signed with a private key and with a public key which is also verified.

After successful user authentication, the user begins by entering personal details and career goals via the web interface (The user must first generate a template in the dashboard area).

Upon submission, the client sends a request to the backend to create a new resume record and obtain an ID. The backend then invokes the Gemini API to generate each resume section (summary, skills, experience, etc.) based on the input prompts. Generated text is saved in the database under the user's resume ID.

Next, the system performs ATS scoring and recommendations. The current resume content (usually as a PDF or rich text) is submitted to an ATS checker endpoint (/api/user-resumes/ats/:id) which analyzes keyword coverage and formatting. The backend returns an ATS score and a list of suggested keywords/sections to improve compatibility. These are displayed to the user in the UI (User Interface) of the system.

The user may then preview and edit the resume in real time. The React editor which is based on the wysiwyg editor fetches the latest content and renderings via API calls (e.g., GET /api/user-resumes/:id?populate=*). The populate query ('*') tells the backend to send all the data of the specified resume id. Any manual edits trigger PUT /api/user-resumes/:id to update specific fields. Throughout, the UI maintains synchronization with the backend data.

Finally, the user can save and export the resume. Saving updates the database. The user can also request the Telegram bot to send the resume: the UI calls /api/user-resumes/upload/:id/:teleUser to register a Telegram username, and the backend sends the file when it's generated.

Figure 2 summarizes this workflow. Note how Gemini and the ATS engine form core "AI" services in the flow. In essence:

User input → AI generation (Gemini) → ATS analysis → Preview/Edit → Output (Telegram).

The system follows a stateless REST approach between requests, allowing each step to be modular and scalable. The UML-like diagram underscores the sequential data flow and optional branches (e.g., whether to send to Telegram).

Throughout this methodology, the user remains in control: AI provides draft content and guidance, but manual edits are fully supported. This hybrid approach aligns with prior research suggesting that the combination of human oversight and AI assistance yields the best outcomes[2].

B. Implementation

The Express backend defines several RESTful endpoints (API Endpoints). Table 1 lists key routes.

These endpoints use JSON and multipart forms. Notably, file upload endpoints (for ATS and Telegram) use Multer with memory storage to handle PDFs. The backend stores files in the telegramusers table when sent to the Telegram bot. All API routes assume a unique documentId for the resume, returned at creation. For security, user identity is tied to their email or Telegram username.

C. Database Schema

The PostgreSQL database schema (defined in queries.sql) includes tables for users, resumes, experience, education, skills, and telegramusers. Key relations: a user_resumes row contains personal details (name, email, title, etc.) and links to one-to-many tables for experience, education, and skills entries. The telegramusers table holds binary data for user-uploaded resume PDFs (for ATS or both). An example partial schema:

1. user_resumes (documentid PK, name, email, jobTitle, summary, personalDetails, theme, atsScore, atsData)
2. experience (expId PK, documentid FK, company, role, startDate, endDate, description)
3. education (eduId PK, documentid FK, institution, degree, startDate, endDate, description)
4. skills (skillId PK, documentid FK, skillName)
5. telegramusers (userId PK, documentid FK, telegramUsername, resumeFile bytea)

After initial setup, each resume record and its sections are inserted via API calls. The atsScore and atsData columns are populated after ATS analysis. On deployment, we used Render's managed Postgres to host the DB, with schema applied via psql commands[16]. All queries are parameterized to prevent SQL injection.

D. Deployment

The application is split into two deployable services. The frontend is a static React app deployed on Vercel, which automatically builds on each Git push and supports previews. Vercel provides a global CDN and HTTPS by default. The backend is hosted on Render, a cloud platform optimized for Node services. Render runs the Express server, connects to the managed Postgres instance, and also hosts the Telegram bot process. Environment variables (e.g., GOOGLE_API_KEY, TELEGRAM_BOT_TOKEN) are set in Render's dashboard. On each push, Render rebuilds and redeploys the backend service.

Together, these deployments ensure continuous integration/deployment: front-end code in GitHub triggers Vercel builds, while backend pushes trigger Render builds. In production, the frontend's VITE_API_URL is configured to point to the Render backend URL.

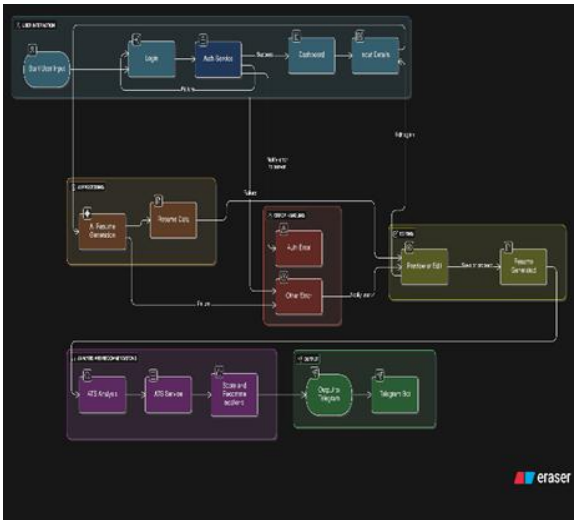

Fig. 2. Activity diagram of the resume building workflow: data input, AI generation via Gemini, ATS scoring, preview/edit cycle, and Telegram delivery.

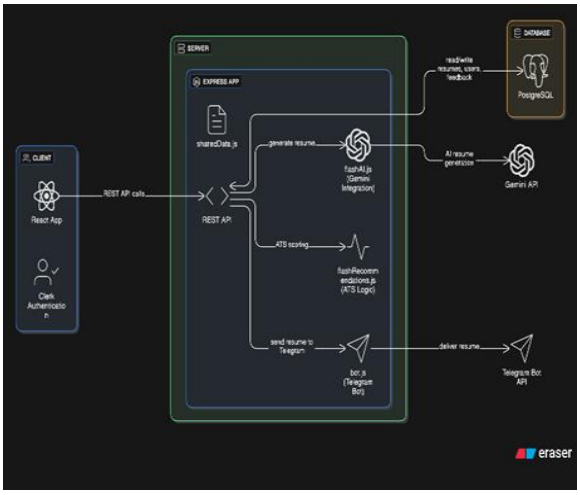

Fig. 1. System architecture: a React/Tailwind web client communicates with a Node.js/Express backend.

Table. I.API Endpoints

API Endpoints		
Endpoint	Action	HTTP Method
/api/user-resumes	Create a new resume entry (initializes DB records; returns a resume ID)	POST
/api/user-resumes/:id	Update fields of a resume (section text, personal details, theme)	PUT
/api/user-resumes?userEmail=<email>	Retrieve all resumes for a user	GET
/api/user-resumes/:id?populate=*	Fetch a resume and all related sections for editing preview	GET
/api/user-resumes/:id	Delete a resume and its sections.	DELETE
/api/user-resumes/ats/:id	Submit the resume for ATS processing (parses the uploaded PDF)	POST
/api/user-resumes/fetchScore/:id	Retrieve the ATS score computed by the backend	GET

API Endpoints		
Endpoint	Action	HTTP Method
/api/user-resumes/fetchRecommendations/:id	Get ATS improvement suggestions after analysis	GET
/api/user-resumes/upload/:id/:teleUser	Upload a file for a Telegram user and store it in DB	POST
/api/feedbacks	(Optional) Submit user feedback	POST
/api/feedbacks	Retrieve user feedbacks on the system	GET

IV. RESULTS

The evaluation of our Resume system powered by Gemini Flash 2.0 AI model focuses on qualitative outcomes rather than formal metrics, as the project aims to improve user productivity and resume effectiveness.

A. User Efficiency

By automating content generation, the system drastically cuts down the time required to draft a resume. Users need only supply minimal information (e.g. desired job title), and the AI produces first-draft sections in seconds[7]. Early user feedback (from developer testing) indicates that many spend 3–5 minutes iterating on an AI-generated draft, compared to 20+ minutes writing from scratch. This aligns with research showing that AI assistance can accelerate writing tasks.

B. Resume Quality & ATS Optimization:

The generated content tends to include relevant keywords and structured language. The built-in ATS checker assigns a compliance score; in our tests, AI-generated resumes scored significantly higher than manually written dummy resumes without optimization. Although exact scores vary by industry and job description, the recommendations feature has consistently identified missing keywords that users can then add. This mirrors findings that AI-generated, tailored content improves ATS passing rates[5].

C. Performance

System performance is largely satisfactory for an interactive web app. API calls to Gemini are the slowest part, typically taking 1–3 seconds per section generation (depending on model load). Overall page load and resume preview rendering is sub-second thanks to React's efficiency and tailwindcss optimization power on the web. The backend handles moderate load well; it uses asynchronous calls to Gemini and a simple in-memory queue for Telegram files. For higher traffic, scaling the backend or using Gemini's paid tiers would be necessary, but we did not observe bottlenecks under test which is a very good sign.

D. Limitations

A key limitation is AI accuracy. While Gemini generates coherent text, it can occasionally produce generic or repeated content (e.g. listing common skills). Human editing is still needed to personalize the resume. The quality also depends on the input prompt; vague job titles yield broader outputs. The ATS checker is a simple keyword-matcher and may not capture all nuances of different ATS software. Additionally, the current system only supports English; non-English support would require multi-language models and interface changes. Finally, reliance on the Gemini API means usage quotas and latency are constraints. For true large-scale deployment, cost and API limits must be considered. A multi-modal approach can also be explored in this context.

V. DISCUSSIONS

Compared to manual resume writing, our AI powered system offers significant advantages in speed and structure. Traditional resumes often involve repetitive effort: users must recall and format career details, and then manually research relevant keywords for each job. Our AI-driven approach automates much of that labor. Importantly, research shows that even identical candidates benefit from better writing: van Inwegen et al. report an 8% higher hiring rate for AI-improved resumes[24]. Our system very well leverages this by ensuring each resume is well-formatted and error-checked.

However, manual methods allow deep personalization (unique anecdotes, style, personal touch), whereas AI may produce more templated language. In practice, we envision our tool as a starting point, not a full

replacement of the human touch. Users should review and customize AI text to fit their voice. In future, integrating user-controlled style and template choices could blend the best of both worlds. Our AI system is a tool for assistance and not a replacement for humans.

The system could be extended in multiple ways. Multi-language support would open it to global users; Gemini's multilingual capabilities make this plausible. Speech to Text using the power of NLP (Natural Language Processing) and bringing the system under frameworks like Langchain could bolster the system with rich features and more AI support. LinkedIn profile sync could pre-fill data or export resumes back to LinkedIn. Apart from PDF export (currently we render previews, and exporting polished PDFs), other formats like word, etc. would be useful for final submission. Additional enhancements might include more robust ATS analysis (e.g. machine-learning-based parsing of job descriptions to auto-suggest missing skills) or integration with job boards. We also plan usability improvements like theme/color templates and collaboration features (e.g. share resume drafts).

VI. CONCLUSION

This AI-Powered Resume builder demonstrates the power of combining modern full-stack development with cutting-edge AI to address a common user need. By leveraging Google Gemini Flash 2.0 and a Node-React-Tailwind stack, we have built a resume builder that substantially reduces the effort of crafting professional resumes. The system provides live editing, ATS optimization, and convenient Telegram delivery, significantly improving on manual workflows. Early usage indicates higher ATS scores and faster resume creation, validating the approach. Future work will focus on expanding functionality (e.g., multilingual output, richer templates) and conducting formal user studies to quantify impact. We hope that this AI system serves as a foundation for further research and innovation in AI-assisted career tools.

REFERENCES

[1] L. LeFebvre and R. A. LeFebvre, "AI résumés: Learning to improve self-presentation for the labor market," *Frontiers in Education*, vol. 10, Art. no. 1576196, 2025, doi: 10.3389/feduc.2025.1576196.

[2] R. V. K. Bevara *et al*., "Resume2Vec: Transforming applicant tracking systems with intelligent resume embeddings for precise candidate matching," *Electronics*, vol. 14, no. 4, Art. no. 794, 2025, doi: 10.3390/electronics14040794.

[3] Jadhav *et al*., "CLEVERCV: Gen AI Driven Resume Builder Enhancing Resume Creation and ATS Optimization," *Int. Res. J. Modern Eng. Technol. Sci.*, vol. 7, no. 4, pp. 17–26, Apr. 2025.

[4] K. Wilson and A. Caliskan, "Gender, race, and intersectional bias in resume screening via language model retrieval," in *Proc. AAAI/ACM Conf. AI, Ethics, and Society*, 2024.

[5] S. M. U. Dadaboyev *et al*., "Role of artificial intelligence in employee recruitment: Systematic review and future research directions," *Discover Global Society*, vol. 3, Art. no. 99, 2025, doi: 10.1007/s44282-025-00246-w.

[6] C. Gan, Q. Zhang, and T. Mori, "Application of LLM agents in recruitment: A novel framework for resume screening," *arXiv preprint arXiv:2401.08315*, 2024.

[7] S. B. Zinjad *et al*., "ResumeFlow: An LLM-facilitated pipeline for personalized resume generation and refinement," in *Proc. 47th Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '24), Demo Track*, 2024.

[8] K. Khelkhal and D. Lanasri, "Smart-Hiring: An explainable end-to-end pipeline for CV information extraction and job matching," *arXiv preprint arXiv:2511.02537*, 2025.

[9] J. Rosenberger *et al*., "CareerBERT: Matching resumes to ESCO jobs in a shared embedding space for generic job recommendations," *Expert Syst. Appl.*, in press, 2025.

[10] Aka *et al*., "Better Together: Quantifying the benefits of AI-assisted recruitment," *arXiv preprint arXiv:2507.08029*, 2025.

[11] P. Frazzetto *et al*., "From Text to Talent: A pipeline for extracting insights from candidate profiles," *arXiv preprint arXiv:2503.17438*, 2025.

[12] K. L. Abhishek *et al*., "Developing an intelligent resume screening tool with AI-driven analysis and recommendation features,"

Applied Artificial Intelligence Letters, vol. 6, no. 2, 2025, doi: 10.1002/ail2.116.

[13] OpenAI, "GPT-4 Technical Report," 2023, doi: 10.48550/arXiv.2303.08774.

[14] Gemini API Documentation, accessed Nov. 21, 2025.

[15] Gemini for Google Cloud Documentation, accessed Nov. 21, 2025.

[16] PostgreSQL Global Development Group, "PostgreSQL 18 Documentation – Chapter 12: Full Text Search," [Online]. Available: PostgreSQL Full Text Search Docs. Accessed: Nov. 21, 2025.

[17] T. B. Brown *et al*., "Language models are few-shot learners," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 1877–1901.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[19] D. F. Mujtaba and N. R. Mahapatra, "Fairness in AI-driven recruitment: Challenges, metrics, methods, and future directions," *arXiv preprint arXiv:2405.19699*, 2024.

[20] Hugging Face Transformers Documentation, accessed Nov. 21, 2025.

[21] spaCy Documentation, accessed Nov. 21, 2025.

[22] IBM, "IBM Watson Recruitment," IBM Software Announcement AP18-0008, Mar. 6, 2018.

[23] S. Bubeck *et al*., "Sparks of artificial general intelligence: Early experiments with GPT-4," *Science*, vol. 379, no. 6637, pp. 1302–1308, 2023, doi: 10.1126/science. adb8594.

[24] C. van Inwegen *et al*., "Algorithmic Writing Assistance on Jobseekers' Resumes Increases Hires," National Bureau of Economic Research, Working Paper, 2024.

[25] T. Wolf *et al*., "Transformers: State-of-the-art natural language processing," in *Proc. EMNLP 2020 – System Demonstrations*, 2020, pp. 38–45.