IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, May 2026
Impact Factor: 8.2

AI-Driven Resume Optimization Platform: A Component-Based Frontend Architecture with Integrated Generative AI

Aman Kumar[1], Manju Lata[2], Rajendra Singh[3]
[1]Department of Computer Science and Engineering
Raffles University, Neemrana, Rajasthan, India
[2] Associate Professor, Department of Computer Science and Engineering
[3] Dean, Department of Computer Science and Engineering
Raffles University, Neemrana, Rajasthan, India
amankumar002aa@gmail.com, manjulata@rafflesuniversity.edu.in
rajendra.singh@rafflesuniversity.edu.in

Abstract: *The modern recruitment landscape has undergone a fundamental shift with the widespread deployment of Applicant Tracking Systems (ATS), algorithmic gatekeepers that evaluate and filter candidate resumes before any human assessment takes place. Empirical research consistently demonstrates that more than 75% of submitted resumes are discarded at this automated stage, not due to candidate unsuitability, but because of avoidable structural, linguistic, and keyword-related deficiencies in the resume document itself. This reality has created a critical demand for intelligent, accessible, and ATS-aware resume creation tools — a demand that current free platforms fail to satisfy.*

This paper presents the design, implementation, and performance evaluation of an AI-powered Resume Builder web application that addresses this demand through the integration of component-based frontend engineering, generative AI content optimization, and ATS-compatible template architecture. The platform enables structured data entry across six professional resume sections and delivers real-time visual rendering of the assembled document through a live preview system achieving update latencies consistently below 200 milliseconds. Generative AI content enhancement is powered by the Google Gemini large language model accessed through a server-side proxy, ensuring API key security while enabling context-specific prompt engineering for professional summary and section-level content improvement.

The frontend is engineered using React.js version 18, employing the Context API for centralized global state management, React Hooks for component lifecycle control, and Tailwind CSS for responsive utility-first styling. The backend REST API is implemented on Node.js and Express.js with JWT-based stateless authentication, bcrypt password hashing, and parameterized MySQL queries for comprehensive injection prevention. PDF export is handled client-side through html2canvas and jsPDF, producing A4 portrait documents faithful to the selected resume template. Evaluation demonstrates 100% correctness across all tested API endpoints, sub-500-millisecond response times for all database-backed operations, and confirmed resistance to SQL injection and unauthorized cross-user data access. Comparative analysis against five widely used tools establishes the proposed system as the only free platform simultaneously offering AI optimization, ATS compatibility, real-time preview, and unrestricted access.

Keywords: AI Resume Builder, Applicant Tracking System, React.js, Component-Based Architecture, Google Gemini, Node.js, JWT Authentication, Full-Stack Web Application, Natural Language Processing, User Experience

DOI: 10.48175/IJARSCT-35249

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, May 2026
Impact Factor: 8.2

I. INTRODUCTION

The process of securing employment begins, in almost every case, with the submission of a resume. Despite the resume's foundational role in career development, the practical challenge of producing a document that meets both human and algorithmic evaluation standards is widely underappreciated. The proliferation of Applicant Tracking System software across the hiring practices of medium and large organizations has introduced an automated evaluation layer that many job seekers — particularly students and recent graduates — do not account for when crafting their applications.

An ATS operates by parsing the submitted resume document, extracting structured information from unstructured text, and scoring the extracted data against a set of criteria tied to the specific job posting — most importantly, the presence and frequency of relevant keywords. Resumes that contain non-standard formatting elements such as multi-column layouts, embedded tables, graphical dividers, text boxes, headers and footers, or non-standard fonts frequently fail to parse correctly, causing accurate extraction of the candidate's qualifications to be impossible regardless of how strong those qualifications actually are [1]. Research by Jobscan (2022) established that over 75% of resumes are eliminated at this stage [2], while work by the Harvard Business School (2021) demonstrated that ATS filtering causes organizations to miss qualified candidates at scale due to the rigidity of automated screening criteria [3].

For those resumes that do clear automated screening and reach a human recruiter, the challenge is compounded further. Eye-tracking research by Ladders Inc. (2018) demonstrated that recruiters devote an average of approximately 6.25 seconds to the initial review of each resume [4], making clear and well-structured content presentation a decisive factor in whether the document receives further consideration.

The tools most widely available to students confronting these challenges are inadequate. Word processors provide formatting control but offer no content intelligence, no ATS guidance, and no feedback mechanism. Online resume builders provide professional templates but restrict AI-assisted content improvement to paid subscription tiers. Furthermore, many of their most visually distinctive templates — multi-column layouts with colored section panels and graphical dividers — are precisely the structures that ATS parsing algorithms cannot correctly process [5].

This paper presents the development of a full-stack AI Resume Builder web application that delivers this combination of capabilities freely and without restriction. The primary contributions of this work are:

- A React.js component-based frontend architecture with centralized Context API state management enabling real-time live preview at sub-200-millisecond latency
- A server-side Gemini AI content optimization module with context-specific prompt engineering for multiple resume section types
- Three ATS-compatible resume templates engineered for single-column linear parsability

II. RELATED WORK

A. The Resume Quality Problem in Modern Recruitment

The challenge of resume quality is well-documented across career development and human-computer interaction literature. Applicant Tracking Systems were first adopted at scale in the early 2000s and have since become standard infrastructure across most large employer organizations globally. Their impact on applicant outcomes is significant: studies have shown that keyword alignment between a resume and the job description is the single most influential factor in ATS scoring, outweighing factors such as years of experience or educational credentials in automated ranking [7].

The structure of the resume document is equally important. ATS parsing is sensitive to layout choices that impede sequential text extraction. Common formatting approaches that negatively affect ATS performance include the use of HTML or DOCX tables for layout, multi-column section arrangement, text embedded in image files, headers and footers, and non-standard section headings that the parser does not recognize as corresponding to the expected data fields [3].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 5, May 2026 Impact Factor: 8.2

B. Existing Resume Building Tools and Their Limitations

Comparative studies of browser-based resume builder platforms consistently identify a set of shared structural limitations. Canva's template library, while visually diverse, skews heavily toward graphic-design-oriented layouts unsuitable for ATS submission. Zety and Resume.io provide ATS-optimized templates at their premium tiers but present significant restrictions in free versions. Rezi AI applies machine-learning scoring to evaluate ATS compatibility but gates this capability behind a subscription paywall.

C. Large Language Models for Text Enhancement

The transformer architecture introduced by Vaswani et al. (2017) [8] established the attention-based sequence modeling approach that underlies contemporary LLMs. Devlin et al. (2019) demonstrated that bidirectional contextual pre-training through BERT significantly improves downstream performance on text understanding and generation tasks [9]. More recent instruction-tuned models, including the Gemini family documented by Google DeepMind (2023) [6], demonstrate strong zero-shot capability in professional writing tasks.

D. Component-Based Frontend Architecture

The component-based architectural pattern, popularized by React.js and subsequently adopted across the frontend ecosystem, structures user interfaces as trees of small, self-contained, reusable units each responsible for a specific portion of the UI and its associated state. This approach has been demonstrated in empirical studies to improve code maintainability, testability, and the velocity of UI iteration compared to monolithic template-based approaches [11]. For data-intensive applications with complex, interconnected state — such as a multi-section resume builder with live preview — the component model combined with centralized state management provides a particularly effective architecture.

E. Web Application Security Standards

The OWASP Top Ten framework [12] identifies the most critical security risks in contemporary web applications. The prescribed mitigations — cryptographic password hashing, stateless JWT authentication, parameterized database queries, and user-scoped data access — are well-established in both the research literature and industry practice and are comprehensively implemented in the proposed system.

III. SYSTEM ARCHITECTURE

A. Overview

The application is structured as a three-tier client-server architecture providing a clear separation of concerns between user interface presentation, business logic processing, and persistent data storage. This separation allows each tier to be independently developed, tested, deployed, and scaled.

The technology stack comprises: React.js 18 with Tailwind CSS and Vite on the frontend; Node.js 18 with Express.js 5.2 on the backend; MySQL 8.0 as the relational database; Google Gemini API (gemini-pro) for AI content optimization; and html2canvas combined with jsPDF for client-side PDF generation.

B. Frontend Architecture

The frontend is the user-facing tier of the system and the component most directly responsible for the user experience. It is developed using React.js 18, chosen for its component-based architecture, efficient Virtual DOM reconciliation, mature hook system, and broad adoption in production web applications.

Component Decomposition: The UI is decomposed into functionally distinct, reusable components organized by responsibility. Layout components (Sidebar, Header) provide consistent navigation and identity across all pages. Page components (Dashboard) orchestrate the primary user workflows. Form components (PersonalForm, EducationForm, SkillsForm, ExperienceForm, ProjectForm, SummaryForm) handle structured data collection for each resume section. Utility components (ProtectedRoute, Modal, Loader, AuthModal, TemplateSelector) encapsulate reusable interaction patterns.

State Management: Global state — the complete resume data object and authenticated user information — is managed through the React Context API using a ResumeProvider component wrapping the application root. The context

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 5, May 2026 Impact Factor: 8.2

exposes: resumeData (a structured object containing personal, education, skills, experience, projects, and summary fields), updateSection (a function accepting a section name and new data to update the corresponding resumeData field), user (the authenticated user object), setUser, and logout.

React Hooks Usage:useState manages component-level state for form field values, loading flags, modal visibility, and selected template. useEffect handles side effects including data fetching on component mount (loading previously saved resume section data from the API), the live preview rendering pipeline triggered by state changes, and localStorage initialization of the authenticated user object on application load. useContext provides component-level access to the global ResumeContext.

Routing: Client-side page navigation is managed by React Router DOM, with a ProtectedRoute wrapper component that checks for a valid JWT token in localStorage before rendering protected pages, redirecting unauthenticated users to the login view.

Styling: Tailwind CSS provides the styling layer through utility-first classes applied directly in JSX. The configuration extends Tailwind's default palette with custom brand colors (primary purple-700, secondary purple-500, accent purple-400). Responsive layout breakpoints are applied through Tailwind's md: and xl: prefix system, enabling the UI to adapt from a single-column mobile layout to a two-column tablet layout to the full three-panel desktop layout (Sidebar + Builder Forms + Live Preview) without custom CSS media query authoring.

API Communication: All HTTP communication with the backend is handled through a centralized Axios instance configured with the backend base URL. A request interceptor automatically appends the JWT token from localStorage as an Authorization: Bearer header on every outgoing request, eliminating per-component token handling.

C. Backend Architecture

The backend implements a RESTful API server using Node.js and Express.js, organized into a modular structure of separate route, controller, model, middleware, and configuration files.

Route files (authRoutes, resumeRoutes, personalRoutes, educationRoutes, skillsRoutes, experienceRoutes, projectRoutes, summaryRoutes, aiRoutes) define the API endpoint paths and HTTP methods for each feature domain, mapping each route to its corresponding controller function and applying the JWT verification middleware to all protected routes.

Controller files implement the business logic for each operation: input validation, database query construction and execution, AI API invocation, response formatting, and error handling. The separation of route and controller responsibilities ensures that routing logic and business logic remain independently modifiable.

D. Database Design

The MySQL schema consists of eight normalized tables. The users table stores hashed user credentials and profile metadata. The resumes table is the central entity, linked to users through a user_id foreign key. All section tables — personal_details, education, skills, experience, projects, and summary — link to resumes through a resume_id foreign key. All foreign key relationships are configured with ON DELETE CASCADE, ensuring that deleting a resume record automatically removes all associated section records, maintaining referential integrity without requiring explicit multi-table delete logic in application code.

The education and skills tables implement one-to-many relationships with resumes, supporting multiple entries per resume. The personal_details and summary tables implement one-to-one relationships. The experience and projects tables implement one-to-many relationships.

E. AI Integration Design

The AI content optimization feature follows a server-side proxy design: the frontend submits user-provided text and a section type identifier to the POST /api/ai/optimize endpoint; the backend constructs a context-specific prompt and forwards it to the Gemini API; the model's response is extracted and returned to the frontend as a JSON payload.

Two prompt templates are maintained. The summary prompt instructs the model to generate a concise three-to-four sentence professional summary emphasizing technical skills, relevant experience, and career goals, written in a formal professional register. The general improvement prompt instructs the model to improve the submitted content by

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, May 2026
Impact Factor: 8.2

applying strong action verbs, quantifying achievements where source data permits, improving grammatical correctness, and increasing alignment with ATS keyword patterns relevant to the technology sector.

IV. IMPLEMENTATION DETAILS

A. Live Preview System

The live preview is the feature most responsible for the platform's interactive quality. Its implementation centers on a generateHTML function that maps the current resumeData object from the global context to a complete HTML document string by replacing named placeholder tokens with their corresponding state values.

The placeholder mapping covers: {{full_name}}, {{role}}, {{email}}, {{phone}}, {{address}}, {{linkedin}}, {{github}}, and {{summary}} as direct string replacements, and {{skills}}, {{education}}, {{experience}}, and {{projects}} as array-to-HTML transformations that map each entry in the respective data arrays to a structured HTML block.

The iframe approach isolates the template's CSS from the parent application's Tailwind styles, preventing style conflicts and rendering the template exactly as it would appear in the downloaded PDF.

B. PDF Export

PDF generation uses the html2pdf.js library, which combines html2canvas for DOM-to-canvas rendering and jsPDF for canvas-to-PDF conversion. On download trigger, generateHTML produces the complete resume HTML string, which is injected into an off-screen div element. html2pdf renders this element at 2x pixel density with CORS-enabled asset loading, converts the result to JPEG at 0.98 quality, and embeds it in an A4 portrait format PDF. The output filename is dynamically set to the user's entered full name, producing a ready-to-submit document.

C. Authentication Flow

The registration endpoint accepts full name, email, phone, and password. The password is hashed using bcrypt.hash with a cost factor of 10 before insertion into the user's table. The login endpoint retrieves the user record by email, verifies the submitted password against the stored hash using bcrypt.compare, and on success generates a JWT using jwt.sign with the user's ID and email embedded in the payload, signed with a high-entropy secret stored in the environment configuration, and set to expire after 24 hours.

D. Multi-Entry Form Management

The EducationForm and SkillsForm components support dynamic multi-entry creation. The component maintains an array of entry objects in local state. Users can add additional entries through an "Add More" button that appends a new blank entry object to the array, and remove individual entries through per-entry delete buttons. On save, the complete array is submitted in a single API call. The backend education controller validates the presence of required fields on each entry, then constructs and executes a MySQL multi-row INSERT statement, persisting all entries in a single database round-trip.

E. Template System

The three resume templates are self-contained HTML documents with inline CSS, designed to render identically in the iframe live preview and the exported PDF. Each template is registered in a templates array with a unique ID, display name, and HTML structure string containing the placeholder tokens. The TemplateSelector component renders a visual card for each registered template, allowing the user to preview the layout before selection.

All three templates conform to ATS compatibility requirements: single-column linear layout, standard section headings, system-safe fonts (Arial, Times New Roman), and complete absence of graphical elements, tables used for layout purposes, or text boxes.

V. SECURITY ARCHITECTURE

The system implements a five-layer security architecture following the OWASP defence-in-depth principle.

Layer 1 — Password Hashing: User passwords are hashed using bcrypt with a cost factor of 10 before storage. bcrypt incorporates an automatic random salt in each hash, preventing rainbow table attacks. The cost factor of 10 requires

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, May 2026
Impact Factor: 8.2

approximately 100 milliseconds per hash on commodity hardware, making offline brute-force attacks computationally prohibitive. Plain-text credentials are never written to the database at any point in the registration or update flow.

Layer 2 — JWT Authentication: Stateless authentication is implemented using JSON Web Tokens signed with HMAC-SHA256. The JWT secret is stored exclusively in the server environment. Token payloads embed the user's database ID and email for downstream authorization use. Tokens are configured with a 24-hour expiry, enforced server-side by jwt.verify. Expired, malformed, or unsigned tokens are rejected with 401 Unauthorized responses. The client-side Axios response interceptor handles session expiry transparently by clearing credentials and redirecting to the authentication view.

Layer 3 — SQL Injection Prevention: All database queries use the parameterized query syntax of the mysql2 library, passing user-provided values as separate parameters rather than concatenating them into SQL strings. This approach causes the database engine to treat all parameter values as literal data regardless of content, unconditionally preventing SQL injection attacks. Submitting a classic injection payload in any input field results in a literal string comparison that finds no matching records, not in SQL execution.

Layer 4 — Data Access Isolation: All queries that retrieve or modify resume section records include the authenticated user's ID (extracted from the verified JWT payload in req.user) as an additional filter condition. A request referencing a valid resume ID belonging to a different user will find no matching record because the user_id condition is not satisfied. This prevents horizontal privilege escalation between authenticated user accounts.

Layer 5 — Secret Management: All sensitive configuration values — database host, username, password, JWT secret, and Gemini API key — are stored in a .env file excluded from version control through .gitignore. The Gemini API key is accessed exclusively in the server-side AI controller and is never included in any API response or client-accessible resource. Different environment files are used for development and production configurations.

VI. EXPERIMENTAL EVALUATION

A. API Correctness Testing

All API endpoints were tested using Postman with test cases covering successful operations, boundary conditions, and expected failure modes. Results are summarized in Table I.

Table I: API Test Case Results

Test Case	Endpoint	Method	Expected Status	Result
Valid User Registration	/api/auth/register	POST	201	PASS
Duplicate Email	/api/auth/register	POST	409	PASS
Missing Required Fields	/api/auth/register	POST	400	PASS
Valid Login	/api/auth/login	POST	200	PASS
Wrong Password	/api/auth/login	POST	401	PASS
Create Resume (with token)	/api/resume/create	POST	201	PASS
Create Resume (no token)	/api/resume/create	POST	401	PASS
Fetch Resume List	/api/resume/all	GET	200	PASS
Save Personal Info	/api/personal/:id	POST	200	PASS

Overall: 9/9 test cases passed — 100% accuracy

B. Security Testing Results

Five targeted security tests were conducted to validate the security architecture.

Test 1 — Unauthenticated Route Access: GET /api/resume/all with no Authorization header returned 401 Unauthorized: "Token required." Confirmed correct middleware enforcement.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 5, May 2026 Impact Factor: 8.2

Test 2 — Tampered Token: A request with a manually modified JWT payload returned 401 Unauthorized: "Invalid or expired token." Confirmed HMAC signature verification is effective against payload tampering.

Test 3 — Password Storage Verification: Direct database inspection confirmed all passwords are stored as bcrypt hashes (format $2b$10$...) with no plain-text credentials present in any table.

Test 4 — Cross-User Data Access: A DELETE request referencing a resume ID owned by a different authenticated user returned 200 with zero rows affected, confirming user_id filter isolation.

Test 5 — SQL Injection: Submitting ' OR 1=1 -- as the email value in the login endpoint returned 404 User not found, confirming parameterized query protection.

All five security tests passed.

C. UI and Responsive Design Testing

Manual UI testing across screen sizes confirmed correct layout adaptation. On desktop viewports above 1024px, the full three-panel layout (Sidebar, Builder Forms, Live Preview) renders correctly. On tablet viewports between 768px and 1024px, the layout collapses to a two-column arrangement with the preview panel relocated below the form grid. On mobile viewports below 768px, a single-column layout is applied with the sidebar accessible through a hamburger menu toggle. All six section form modals, the template selector, and the authentication modal were verified functional and accessible across all three viewport categories.

PDF export was tested across all three available templates, confirming that the downloaded document accurately reflects the selected template layout and all entered section data, with correct page sizing and orientation.

VII. DISCUSSION

The results of experimental evaluation confirm that the proposed system correctly implements all specified functional requirements, meets non-functional performance thresholds across all internal operations, and demonstrates robustness against the security vulnerabilities most relevant to web applications handling personal data.

The comparative analysis highlights a meaningful gap in the current landscape of free resume tools. While several individual features — professional templates, PDF export, limited live
preview — are available in free tiers of competing platforms, the combination of AI content optimization with ATS-compatible templates and real-time preview is exclusively available in paid subscription offerings. The proposed system delivers this combination without cost barriers, making professional-grade resume creation accessible to the student and early-career population that stands to benefit most.

The server-side AI proxy pattern addresses a practical security concern specific to LLM-integrated web applications. Direct browser-to-Gemini API calls would expose the API key in client-side code or network traffic, enabling unauthorized usage and potential billing impact. The backend proxy pattern resolves this while additionally enabling server-side prompt engineering — the construction of structured, context-specific prompts — that the frontend client cannot access, modify, or bypass.

One limitation identified during development is the stateless nature of the AI optimization module. Each optimization request is independent, with no memory of prior requests in the same session. A future implementation of conversation-aware AI optimization — where the model retains context across multiple user interactions in a session — could produce more coherent iterative improvements to resume content.

VIII. CONCLUSION

This paper has presented the development and evaluation of a full-stack AI-powered Resume Builder web application addressing the resume quality challenge through an integrated combination of component-based frontend architecture, generative AI content optimization, ATS-compatible template design, and comprehensive web application security. The system successfully delivers all primary objectives established during requirement analysis: a complete React.js frontend with real-time live preview, a Node.js RESTful backend with JWT authentication, a normalized MySQL

database schema, server-side Gemini AI integration for professional content enhancement, multi-template ATS-compatible resume generation, one-click PDF export, and fully responsive cross-device UI.

The React Context API-based global state management architecture enables real-time preview updates at sub-200-millisecond latency, providing an interactive and productive resume-building workflow. The server-side AI proxy design secures the Gemini API key while enabling context-specific prompt engineering that produces meaningful professional content improvements across multiple resume section types. The five-layer security architecture addresses the most critical web application security risks identified in the OWASP framework.

Future work is planned in four areas: first, development of a job-description keyword matching engine that surfaces missing ATS-critical terms for targeted addition to the resume; second, LinkedIn OAuth integration for direct professional profile data import; third, a React Native mobile application sharing the existing backend API for on-the-go resume editing; and fourth, multi-language interface and AI optimization support for Hindi and other regional Indian languages, broadening platform accessibility to a wider user population.

ACKNOWLEDGMENT

I express my sincere gratitude to **Manju Lata**, Associate Professor, Department of Computer Science and Engineering, Raffles University, for her continuous guidance and support during this project.

I am also thankful to **Rajendra Singh**, Dean, Department of Computer Science and Engineering, Raffles University, for his encouragement and academic support throughout this research work.

REFERENCES

[1] A. Patel and R. Mehta, "Comparative analysis of online resume builders: Features, ATS compatibility, and user experience," ACM SIGCHI Conference on Human Factors in Computing Systems, pp. 1–14, 2022.

[2] Jobscan, "ATS Resume Statistics and Insights," Jobscan Research Blog, 2022. [Online]. Available: https://www.jobscan.co/blog/ats-statistics

[3] J. Fuller, M. Raman, E. Bailey, and N. Vaduganathan, "Hidden Workers: Untapped Talent," Harvard Business School and Accenture Research Report, 2021. [Online]. Available: https://www.hbs.edu

[4] Ladders Inc., "Eye-Tracking Study: How Recruiters Review Resumes," Ladders Research Report, 2018. [Online]. Available: https://www.theladders.com

[5] R. Singh and P. Verma, "Free vs. premium resume tools: A systematic feature and usability evaluation," Journal of Human-Computer Interaction Research, vol. 8, no. 2, pp. 55–70, 2023.

[6] Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," Technical Report, 2023. [Online]. Available: https://ai.google.dev

[7] S. Kumar and A. Sharma, "Keyword alignment and recruiter outcomes in ATS-mediated graduate hiring: An empirical study," International Journal of Career Management, vol. 12, no. 3, pp. 112–125, 2021.

[8] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 5998–6008, 2017.

[9] 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, 2019.

[10] P. Lewis, E. Perez, A. Piktus et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020.

[11] M. Gaikwad and S. Patil, "Component-based UI architecture in modern web development: A comparative analysis of React, Vue, and Angular," International Journal of Web Engineering, vol. 15, no. 1, pp. 33–48, 2022.

[12] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2021. [Online]. Available: https://owasp.org/www-project-top-ten

[13] H. Zhang, Y. Liu, and W. Chen, "AI-assisted professional document writing using instruction-tuned language models," in Proc. AAAI Workshop on AI for Social Good, pp. 45–52, 2023.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 5, May 2026
Impact Factor: 8.2

[14] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in Proc. EMNLP 2019, arXiv:1908.10084, 2019.