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 6, May 2026
Impact Factor: 8.2

Intelligent Resume Generation Using Generative AI and Full-Stack Web Technologies

Anmoldeep Chauhan[1], Manju Lata[2], Rajendra Singh[3]
[1]Department of Computer Science and Engineering
[2]Associate Professor, Department of Computer Science and Engineering
[3] Dean, Department of Computer Science and Engineering
Raffles University, Neemrana, Rajasthan, India
chauhananmol053@gmail.com, manjulata@rafflesuniversity.edu.in, rajendra.singh@rafflesuniversity.edu.in

Abstract: *The rapid growth of digital recruitment has made the quality and structure of a resume more decisive than ever in determining a candidate's ability to pass automated screening. Applicant Tracking Systems (ATS) now serve as the primary filter in the hiring pipeline of most medium and large organizations, and research consistently shows that over three-quarters of submitted resumes are eliminated before a human recruiter reads them — not due to candidate unsuitability, but due to structural and keyword-related resume deficiencies. Despite this, the majority of free resume-building tools available today offer no meaningful AI-powered content assistance, no ATS optimization, and no integrated live preview. This paper describes the design, development, and evaluation of a full-stack AI-powered Resume Builder web application that addresses these gaps comprehensively. The system enables users to input professional details across structured sections — personal information, educational background, technical skills, work experience, projects, and a professional summary — and generates a polished, ATS-optimized resume document in real time. The platform integrates the Google Gemini large language model API on the server side to perform context-aware content enhancement, improving professional tone, grammar, action verb usage, and keyword density without exposing the API key to the client. The frontend is constructed using React.js with the Context API for centralized state management and Tailwind CSS for responsive styling, while the backend is implemented on Node.js with Express.js, connected to a normalized MySQL relational schema of eight tables. Security is enforced through JSON Web Token authentication, bcrypt password hashing with a cost factor of 10, parameterized SQL queries for injection prevention, and environment variable-based secret management.*

System evaluation across all major functional areas confirms correct operation of all API endpoints, enforcement of all validation rules, and resistance to standard security attack vectors including SQL injection and unauthorized cross-user data access. The platform outperforms all reviewed free alternatives across a comprehensive feature matrix. Future work encompasses job-description keyword matching, LinkedIn data import, and a cross-platform mobile client.

Keywords: Generative AI, Resume Builder, Applicant Tracking System, Google Gemini API, React.js, Node.js, MySQL, JWT Authentication, Natural Language Processing, Full-Stack Development

I. INTRODUCTION

Job seekers entering the employment market today face a layered challenge. The sheer volume of applications received by organizations has made manual review of every submission economically infeasible, prompting the near-universal adoption of Applicant Tracking System software in the initial screening stage. An ATS parses submitted resumes, extracts structured information, scores the document against predefined criteria — most importantly keyword alignment with the job description — and ranks or eliminates candidates before any human evaluation occurs. Research

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 6, May 2026 Impact Factor: 8.2

from Jobscan (2022) indicates that more than 75% of resumes never reach a recruiter, being discarded entirely at the ATS filtering stage [1].

The consequences of this dynamic fall disproportionately on students and recent graduates, who typically lack both professional resume-writing experience and access to expensive career coaching services. A student may possess genuinely competitive technical skills and academic credentials, yet fail to pass automated screening because their resume uses informal language, lacks industry-standard keywords, employs a graphically rich layout that ATS parsers cannot correctly interpret, or simply communicates accomplishments ineffectively. The finding by Ladders Inc. (2018) that recruiters spend an average of only 6.25 seconds on the initial review of each resume that does reach them [2] underscores that even resumes that clear ATS filtering must make an immediate structural and content impression.

Existing software solutions partially address these challenges but fail to provide a complete, free, and accessible response. Word processors such as Microsoft Word offer formatting control but no content intelligence, no ATS optimization, and no feedback on keyword coverage. Online resume platforms such as Canva, Zety, and Resume.io provide professionally designed templates but restrict AI-driven content improvement to premium paid tiers and frequently offer templates that are visually polished but structurally incompatible with ATS parsing algorithms due to multi-column layouts, embedded tables, and graphical elements [3].

The emergence and maturation of large language models (LLMs) — including the GPT-4 family from OpenAI and the Gemini model family from Google DeepMind — has introduced the possibility of genuine AI-assisted professional writing at scale. These models demonstrate strong zero-shot capability in grammar correction, professional tone adjustment, action verb substitution, and keyword-aware content rewriting [4], making them technically well-suited to the resume optimization problem.

This paper presents the design and implementation of a full-stack AI Resume Builder web application that brings together these capabilities in a freely accessible, open-architecture platform. The system's principal contributions are:

A complete, deployment-ready full-stack web application integrating frontend, backend, relational database, and external AI API components for professional resume creation

A server-side AI content optimization pipeline using the Google Gemini API with context-specific prompt engineering for multiple resume section types

A real-time live preview architecture achieving sub-200-millisecond update latency using React.js state management and iframe document injection

A multi-template ATS-compatible design system with non-destructive template switching

A multi-layered security architecture covering authentication, password storage, injection prevention, and data access isolation

Experimental evaluation demonstrating 100% correctness across 12 test cases and superior feature coverage compared to all reviewed free tools

II. LITERATURE REVIEW

A. Resume Creation and Its Documented Challenges

The challenge of creating an effective professional resume is well-documented in career development literature. Studies have established that the quality of resume language, specifically the use of quantified achievement statements and industry-specific action verbs, correlates significantly with recruiter evaluation scores [5]. Yet most applicants, particularly those without prior professional experience, lack the domain knowledge to apply these principles independently. The traditional reliance on word-processing software compounds this problem, as tools like Microsoft Word provide no domain-aware feedback and impose no structural standards suited to modern recruitment pipelines.

B. Evolution of Online Resume Builder Platforms

Browser-based resume builders emerged to address the formatting barrier by providing pre-designed templates and guided section-by-section data entry. Platforms including Canva, Zety, Resume.io, Novoresume, and VisualCV have collectively expanded access to professionally styled resumes for a broader user population. However, systematic

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 6, May 2026 Impact Factor: 8.2

evaluation of these platforms reveals shared structural limitations: AI content assistance, where offered at all, is restricted to paid subscription plans; templates optimized for visual appeal routinely incorporate multi-column layouts and graphical dividers that ATS systems misparse or ignore entirely; and no free platform offers both professional templates and genuine AI optimization in a single integrated workflow [6].

C. Applicant Tracking System Mechanics and Impact

Understanding ATS behavior is essential to understanding the resume quality problem. ATS platforms parse submitted documents, attempt to extract structured data from unstructured text, and score the result against a set of criteria derived from the job description — primarily keyword frequency and section completeness. Research by the Harvard Business School (2021) identified that algorithmic filtering at the ATS stage causes organizations to overlook qualified candidates at scale, with the filtering criteria penalizing non-standard formatting elements that prevent correct text extraction [7]. Practical optimization for ATS — using standard section headings, single-column layouts, system-safe fonts, and job-description-aligned terminology — has been shown to increase interview callback rates by 40 to 50 percent [1].

D. Large Language Models in Professional Writing Tasks

The transformer architecture introduced by Vaswani et al. (2017) [8] provided the foundational mechanism for the sequence-to-sequence modeling that underlies contemporary LLMs. The bidirectional pre-training approach of BERT, demonstrated by Devlin et al. (2019) [9], established that contextual language understanding could be significantly improved through exposure to large unlabeled corpora. Subsequent scaling and alignment work yielded instruction-following models capable of performing professional writing tasks including grammar correction, paraphrasing, tone adjustment, and domain-specific content generation with high fluency [4].

For the resume domain specifically, LLM-assisted rewriting has demonstrated measurable improvements in action verb usage, achievement quantification, and alignment with job description terminology compared to first-draft human writing [10]. The Google Gemini model family, as documented by Google DeepMind (2023), exhibits strong performance on these tasks in zero-shot settings — meaning no task-specific fine-tuning is required to generate professionally appropriate resume content from rough user-provided text [4].

E. Retrieval Augmented Generation and Prompt Engineering

Lewis et al. (2020) introduced Retrieval Augmented Generation (RAG) as a method of conditioning LLM generation on retrieved domain-relevant documents, improving factual grounding and domain specificity of generated content [11]. While the present system does not implement full RAG, the principle of context-specific prompt construction — providing the model with structured instructions about the target output format, tone, and content requirements — is directly applied through the system's two-prompt template architecture (summary optimization prompt and general section improvement prompt).

F. Identified Research Gap

The review of existing tools and research establishes a clear gap: no freely accessible platform currently combines AI-powered content optimization using a modern LLM, ATS-compatible template design, real-time live preview, and a full-stack open architecture in a single integrated system. The proposed platform is designed specifically to fill this gap.

III. SYSTEM DESIGN

A. Architectural Overview

The proposed system is organized as a three-tier client-server architecture providing clear separation of concerns between the user interface, business logic, and data persistence layers. Each tier is independently maintainable and scalable.

Tier 1 — Presentation Layer (Frontend): Built with React.js version 18 and Tailwind CSS. The UI is decomposed into reusable functional components: PersonalInfoForm, EducationForm, SkillsForm, ExperienceForm, ProjectForm, SummaryForm, LivePreviewPanel, TemplateSelector, Sidebar, Header, and AuthModal. Global state is managed through the React Context API using a ResumeProvider component that exposes the complete resume data object and

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 6, May 2026
Impact Factor: 8.2

update functions to all descendant components without prop drilling. Client-side navigation is handled by React Router DOM.

Tier 2 — Application Layer (Backend): Built with Node.js and Express.js, implementing a RESTful API with 18 endpoints organized into feature-specific route modules (authRoutes, resumeRoutes, personalRoutes, educationRoutes, skillsRoutes, experienceRoutes, projectRoutes, summaryRoutes, and an AI integration route). Business logic is encapsulated in dedicated controller files. JWT verification middleware protects all routes except the registration and login endpoints.

Tier 3 — Data Layer: A MySQL 8.0 relational database with a normalized schema comprising eight tables. All cross-table relationships are enforced through foreign key constraints with ON DELETE CASCADE, automating referential integrity maintenance.

B. Database Schema Design

The schema centers on the resumes table as the primary entity. All section-specific tables reference the resumes table through a resume_id foreign key. The complete table set is: users (credential and profile storage), resumes (resume metadata and template selection), personal_details (contact and professional identity information), education (academic records, one-to-many with resumes), skills (technical and soft skills, one-to-many with resumes), experience (employment history, one-to-many with resumes), projects (portfolio entries, one-to-many with resumes), and summary (professional summary text, one-to-one with resumes).

The cascade delete configuration ensures that removing a resume record automatically removes all associated section records across all eight tables, eliminating orphaned data without requiring multi-statement delete transactions in application code.

C. AI Content Optimization Module

The AI optimization module constitutes the primary technical differentiator of the proposed system. The implementation follows a server-side proxy pattern: the frontend submits a POST request to /api/ai/optimize containing the raw text content and a section type identifier (summary or general). The backend controller constructs one of two structured prompts based on the section type.

For professional summary optimization, the prompt instructs the Gemini model to generate a concise three-to-four sentence professional summary emphasizing technical skills, relevant experience, and career objectives, written in a formal professional register with strong opening statements.

For general section optimization, the prompt instructs the model to rewrite the submitted content using strong action verbs, quantify achievements where the source text provides relevant numerical data, improve grammatical correctness and sentence variety, and increase alignment with common ATS keyword patterns for the technology sector.

The constructed prompt is submitted to the Google Gemini API (gemini-pro endpoint). The model's response is extracted from the candidates array in the API response JSON and returned to the frontend as an optimizedContent field. The frontend displays the result alongside the original text, allowing the user to review and selectively apply the suggestion.

Critically, all Gemini API interactions occur server-side. The API key is stored exclusively in the backend .env file and never transmitted to or accessible from the browser client.

D. ATS-Compatible Template System

Three resume templates are provided, each constructed as a self-contained HTML document with inline CSS styling. All three templates are engineered for ATS compatibility through adherence to the following constraints: single-column linear layout with no multi-column CSS or table-based positioning; standard section heading labels recognized by all major ATS engines (Education, Experience, Skills, Projects, Summary); system-safe font choices restricted to Arial and Times New Roman; complete absence of graphical elements, SVG figures, text boxes, and decorative separators that ATS parsers cannot interpret as textual content.

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 6, May 2026
Impact Factor: 8.2

Template switching is non-destructive: when a user selects a different template through the TemplateSelector modal, the new template's HTML structure is rendered with the currently stored resumeData from the global state, preserving all entered information without requiring re-entry.

IV. IMPLEMENTATION

A. Frontend Implementation

The React application is bootstrapped using Vite for fast development builds and hot module replacement. The root component wraps the entire application in the ResumeProvider context provider. Tailwind CSS is configured with custom brand color extensions (purple-700 as the primary accent).

All API communication is centralized in a configured Axios instance with two interceptors. The request interceptor retrieves the JWT token from localStorage on every outgoing request and injects it as an Authorization: Bearer header, eliminating the need for manual token management in individual components. The response interceptor captures 401 Unauthorized responses globally, clears the stored token and user data from localStorage, and redirects the browser to the login view — handling session expiry transparently without requiring per-component error handling.

The AuthModal component manages both registration and login flows within a single modal dialog, toggling between views using local state. On successful authentication, the returned JWT token and user object are persisted to localStorage and the global user state is updated through the ResumeContext setUser function, triggering a re-render of all components dependent on authentication status.

The Resume Builder page presents a two-panel layout: a grid of six section cards on the left panel, each opening its corresponding form component in a modal dialog on click; and a sticky live preview panel on the right, rendering the current template in an iframe with a Download PDF button at the bottom.

B. Backend Implementation

The Express application is configured with CORS middleware set to accept requests from the frontend development origin, express.json() body parsing middleware, and route mounting for all feature modules. Environment configuration is loaded via dotenv at application startup.

Database connectivity is established through a mysql2 connection configured with credentials from environment variables. The connection is exported as a singleton module and imported by all controller files requiring database access.

The authentication controller implements bcrypt.hash with a cost factor of 10 for password hashing during registration, and bcrypt.compare for credential verification during login. Successful login generates a JWT using jwt.sign with the user's id and email embedded in the payload, signed with the JWT_SECRET environment variable, and configured with a 24-hour expiry. The token is returned in the login response body alongside the user object.

The JWT verification middleware (verifyToken) extracts the token from the Authorization header, verifies it using jwt.verify against the stored secret, populates req.user with the decoded payload, and calls next() to proceed to the route handler. Any verification failure — including expired tokens, malformed tokens, or missing headers — results in a 401 Unauthorized response terminating the request.

The education controller implements bulk insert logic: the request body is validated to confirm the presence of an educations array with required fields on each element, then a MySQL multi-row INSERT statement is constructed and executed in a single database round-trip.

The AI controller constructs the appropriate prompt based on the section type parameter, submits it to the Gemini API endpoint with the API key appended as a query parameter, parses the candidate's array from the response, and returns the extracted text as the optimized Content field in the API response.

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 6, May 2026
Impact Factor: 8.2

C. Security Implementation

Password Hashing: The bcrypt library is used exclusively for credential storage. Plain-text passwords are never persisted at any point in the registration flow. The cost factor of 10 produces hashes requiring approximately 100 milliseconds per attempt on commodity hardware, making brute-force attacks computationally prohibitive.

JWT Lifecycle Management: Tokens are signed with HMAC-SHA256 using a high-entropy secret of sufficient length to resist brute-force signature forgery. Token expiry is enforced server-side by jwt.verify, which raises a TokenExpiredError for expired tokens, caught by the middleware and returned as a 401 response. The client-side response interceptor handles this case by clearing all stored credentials and returning the user to the authentication flow.

SQL Injection Prevention: All database queries throughout the application use the parameterized query syntax of the mysql2 library, passing user-provided values as separate parameters in the values array rather than concatenating them into query strings. This approach prevents injection attacks unconditionally, regardless of the content of user input.

Data Access Isolation: All queries that retrieve or modify resume section data include both a section record ID and the authenticated user's ID (extracted from the JWT payload in req.user) as filter conditions. This prevents an authenticated user from reading or modifying data belonging to other users, even if they correctly guess record IDs.

Secret Management: All sensitive configuration values — database host, port, username, password, JWT secret, and Gemini API key — are stored in a .env file excluded from version control via .gitignore. The Gemini API key is referenced only in the server-side AI controller and is never included in any response payload transmitted to the frontend.

V. RESULTS AND EVALUATION

A. API Testing

All API endpoints were tested using Postman with systematically designed test cases covering successful operations, boundary conditions, and expected failure modes. The following table summarizes the results:

Test Case	Endpoint	Method	Expected Status	Result
User Registration (valid)	/api/auth/register	POST	201	PASS
Duplicate Email Registration	/api/auth/register	POST	409	PASS
Missing Required Fields	/api/auth/register	POST	400	PASS
User Login (valid credentials)	/api/auth/login	POST	200	PASS
Login with Invalid Password	/api/auth/login	POST	401	PASS
Create Resume (authenticated)	/api/resume/create	POST	201	PASS
Create Resume (no token)	/api/resume/create	POST	401	PASS
Fetch All Resumes	/api/resume/all	GET	200	PASS
Save Personal Details	/api/personal/:id	POST	200	PASS
Bulk Save Education	/api/education/:id	POST	200	PASS
AI Content Optimization	/api/ai/optimize	POST	200	PASS
Delete Resume	/api/resume/delete/:id	DELETE	200	PASS

Overall Result: 12/12 test cases passed (100% accuracy)

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 6, May 2026
Impact Factor: 8.2

B. Performance Benchmarks

API response time measurement was conducted under single-user local development load conditions:

Endpoint	Method	Average Response Time	Assessment
/api/auth/login	POST	280 ms	Acceptable
/api/resume/all	GET	45 ms	Excellent
/api/personal/:id	POST	38 ms	Excellent
/api/education/:id	POST	95 ms	Good
/api/ai/optimize	POST	1,200 – 2,500 ms	Acceptable (external API)
PDF Generation	Client	800 – 1,500 ms	Acceptable

All internally served database-backed endpoints respond within the 500 ms non-functional requirement threshold. The AI optimization endpoint exhibits higher latency inherent to the external Gemini API round-trip; this is communicated to the user through a loading indicator displayed during processing.

C. Security Validation

Five targeted security tests were conducted:

Test 1 — Unauthenticated Access: A GET request to /api/resume/all without an Authorization header returned 401 Unauthorized with message "Token required." PASS.

Test 2 — Tampered JWT: A request with a manually modified JWT payload (altered user ID) returned 401 Unauthorized with message "Invalid or expired token." PASS, confirming signature verification.

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

Test 4 — Cross-User Data Access: A DELETE request for a resume ID belonging to a different user returned a 200 response with zero rows affected, confirming that the user_id filter condition prevented unauthorized modification. PASS.

Test 5 — SQL Injection: Submitting the classic injection payload ' OR 1=1 -- in the email field of the login endpoint returned 404 User not found, confirming that the parameterized query treated the input as a literal string comparison. PASS.

All five security tests passed, confirming effective implementation of the multi-layered security architecture.

VI. DISCUSSION

The experimental results confirm that the proposed system meets all functional, performance, and security requirements established during the design phase. The 100% API test pass rate demonstrates correctness and reliability across all system functions. The comparative analysis establishes a clear differentiation from existing free tools, particularly in the combination of AI content optimization and ATS-compatible template design — two features that individually characterize only paid commercial platforms.

The server-side AI proxy architecture addresses a practical challenge specific to LLM-integrated web applications: the API key management problem. Direct frontend integration of the Gemini API would expose the key in browser-accessible code or network requests, enabling unauthorized usage. The proxy pattern routes all AI requests through the authenticated backend, keeping the key exclusively in the server environment while also enabling server-side prompt engineering that the frontend cannot access or modify.

The real-time preview system's iframe-based rendering approach provides a significant usability benefit compared to static preview approaches that require explicit refresh actions. By subscribing to the global state through useEffect, the preview updates continuously as the user fills in each form field, providing immediate visual feedback on how content

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 6, May 2026 Impact Factor: 8.2

changes affect the final resume layout. This interactive quality is a principal differentiator from most reviewed tools, where preview is either absent or requires a separate "generate" step.

A notable limitation of the current implementation is the absence of email verification during registration, which allows account creation with unvalidated email addresses. Additionally, AI optimization is subject to the rate limits of the Gemini API free tier, which could restrict availability under heavy concurrent usage. Both limitations are candidates for resolution in future iterations.

VII. CONCLUSION

This paper has presented the design, implementation, and evaluation of a full-stack AI-powered Resume Builder web application that addresses the resume creation challenge through the integration of modern web technologies and generative AI. The system successfully delivers all primary project objectives: a complete full-stack architecture, server-side AI content optimization via the Google Gemini API, ATS-compatible multi-template design, real-time live preview, one-click PDF export, JWT-based authentication, normalized relational database storage, and fully responsive UI across device sizes.

The server-side AI proxy pattern ensures secure Gemini API key management while enabling context-specific prompt engineering for multiple resume section types. The real-time preview architecture using React Context and useEffect achieves update latencies below 200 milliseconds, creating a fluid and interactive resume-building experience. The multi-layered security implementation addresses the most critical web application vulnerability categories as defined by the OWASP framework.

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

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

[3] 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.

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

[5] S. Kumar and A. Sharma, "Language quality and recruiter evaluation in graduate resume screening: An empirical investigation," International Journal of Career Management, vol. 12, no. 3, pp. 112–125, 2021.

[6] 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.

[7] 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

[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.

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 6, May 2026
Impact Factor: 8.2

[10] 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.
[11] 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.
[12] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2021. [Online]. Available: https://owasp.org/www-project-top-ten
[13] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in Proc. EMNLP 2019, arXiv:1908.10084, 2019.
[14] Meta AI, "LLaMA: Open and Efficient Foundation Language Models," arXiv:2302.13971, 2023