Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

AI Powered Research Paper Generator

Shiv Sagar Giri
Dept. of Computer Science & Eng.
Bansal Inst. of Eng. and Tech.
Lucknow, India

Akshay Singh
Dept. of Computer Science & Eng.
Bansal Inst. of Eng. and Tech.
Lucknow, India

Deepesh Singh
Dept. of Computer Science & Eng.
Bansal Inst. of Eng. and Tech.
Lucknow, India

Ram Kailash Gupta
Dept. of Computer Science & Eng.
Bansal Inst. of Eng. and Tech.
Lucknow, India

Abstract **- Creating academic research papers demands significant time, specialized knowledge, and strict adherence to formatting guidelines like IEEE and APA standards. This paper introduces the AI Powered Research Paper Generator, a web-based intelligent platform that automates the creation of properly formatted academic research papers. The system uses a large language model accessed through a serverless edge function setup to produce both written content and compilable LaTeX source code based on user specifications. The frontend utilizes React 18 with TypeScript and Vite, enhanced with Tailwind CSS and shadcn/ui components for styling. The backend operates entirely on Supabase, which handles user authentication, maintains a PostgreSQL database for storing paper history, and runs two edge functions: one for AI-powered paper creation and another for server-side LaTeX-to-PDF conversion using pdflatex. The platform offers a complete workflow from user login and parameter input to final formatted document output, dramatically reducing the manual work required for academic paper creation. The architecture supports future expansion into multi-format exports, plagiarism checking, and citation management features. This paper outlines the system design, component structure, data processing flow, and core technologies while examining the challenges, constraints, and potential development paths for the platform. Keywords Academic paper generation; large language models; LaTeX automation; IEEE formatting; React; Supabase; serverless edge functions; AI writing assistant; document automation; natural language processing.**

Index Terms: Academic paper generation, large language models, LaTeX automation, IEEE formatting, Supabase, serverless computing

I. INTRODUCTION

Writing academic research papers ranks among the most intellectually challenging aspects of scholarly work. Researchers must juggle multiple complex tasks: creating meaningful content, building logical arguments, managing citations, and following precise formatting rules established by journals or conferences such as the Institute of Electrical and Electronics Engineers (IEEE) or the American Psychological Association (APA). Engineering students and novice researchers face

additional hurdles when learning specialized tools like LaTeX, which adds another layer of complexity to an already difficult process. Although platforms like Overleaf have made LaTeX collaboration easier, authors still need to manually develop all paper content. Recent advances in Large Language Models, including GPT-4 and Claude, have shown remarkable abilities in producing extended academic texts, organizing logical structures, and performing contextual analysis. Combining these models with automated document formatting systems offers exciting possibilities for improving academic productivity tools. The AI Powered Research Paper Generator represents a comprehensive, web-based platform that combines user-friendly academic paper setup with an AI backend that can produce IEEE or APA formatted research papers from basic user inputs. The system primarily serves engineering students and beginning researchers who need to create well-structured draft papers without extensive LaTeX knowledge or detailed formatting experience. The platform consists of four main elements: a React-based single-page application frontend using TypeScript, Supabase for user authentication and data storage, a serverless edge function that communicates with an LLM for content and LaTeX creation, and a secondary edge function that converts LaTeX to PDF using pdflatex. These components work together to provide a complete document creation process accessible through any web browser. This paper makes several key contributions: First, it provides a comprehensive architectural overview of a web-based AI-driven academic paper generation platform. Second, it presents a serverless backend design using Supabase Edge Functions for LLM communication and LaTeX processing. Third, it examines the challenges involved in prompt engineering for structured document creation. Fourth, it analyzes current system limitations and proposes a development roadmap for future improvements. The paper structure proceeds as follows: Section II reviews existing work in AI writing assistance and document automation. Section III explains the system architecture. Section IV covers the frontend design. Section V examines the backend and edge function implementation. Section VI discusses the LaTeX editor and compilation process. Section VII addresses system challenges and limitations. Section VIII presents future

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

development plans, and Section IX provides concluding remarks.

II. RELATED WORK

A. AI-Assisted Academic Writing

AI applications in academic writing have expanded rapidly since transformer-based language models became publicly available. Brown et al. demonstrated that GPT-3 could produce coherent, contextually appropriate extended texts, proving that LLM-assisted document creation was technically feasible. Later developments by OpenAI with GPT-4 showed major advances in following instructions and generating structured outputs, making it more reliable for meeting formatting requirements when given detailed prompts. Academic-focused AI applications now include abstract creation, literature review support, and grammar checking. However, combining AI- generated content with formal typesetting systems like LaTeX, especially for specific journal formatting standards, remains largely unexplored. The AI Powered Research Paper Generator fills this void by merging content creation with LaTeX code generation in one unified process.

B. Document Automation Tools

Overleaf has established itself as the leading platform for collaborative LaTeX editing in academic settings, providing real-time compilation and template collections. Yet it functions purely as an editing environment without content generation capabilities. Other tools like Authorea and Typeset.io offer basic template-based generation but lack sophisticated AI integration. While previous systems have used pdflatex for server-side compilation in web applications, none have combined this with LLM-generated source code. Commercial products such as Jenni AI, Scholarcy, and SciSpace provide AI-powered writing assistance, but they produce plain text rather than compilable LaTeX source and don't specifically target IEEE or APA formatting requirements at the structural and markup levels. The AI Powered Research Paper Generator stands apart by creating syntactically correct LaTeX that includes appropriate document class declarations, author sections, structured environments, and bibliography entries following IEEE standards.

C. Serverless Architectures

Serverless computing adoption for web backends has grown significantly with platforms like AWS Lambda, Vercel Functions, and Supabase Edge Functions. These platforms run temporary functions on demand, removing the need for ongoing server maintenance. For AI-powered applications where backend requests may be sporadic but resource-intensive, serverless functions provide cost effectiveness and automatic scaling. Previous research has examined serverless deployments for natural language processing workflows, and the AI Powered Research Paper Generator builds on this approach by incorporating LaTeX compilation within the serverless framework.

III. SYSTEM ARCHITECTURE

The AI Powered Research Paper Generator uses a three-tier client-server architecture modified for serverless backend operation. The three layers include: the client layer, built as a React Single-Page Application; the service layer, implemented through Supabase Edge Functions on the Supabase platform; and the data layer, using Supabase's managed PostgreSQL database and authentication services. Figure 1 shows the system's overall architecture. Client communication with Supabase services occurs over HTTPS using the official Supabase JavaScript client SDK. Authentication tokens are stored client-side through Supabase Auth, and all database operations follow Row-Level Security policies that limit data access to each authenticated user's own records.

The AI Powered Research Paper Generator implements a three- client-server architecture:

- Client Tier: React SPA
- Service Tier: Supabase Edge Functions
- Persistence Tier: PostgreSQL Database

A. Component Summary

TABLE I
SYSTEM COMPONENT SUMMARY

Tier	Technology	Responsibility
Client	React + TypeScript	UI rendering, routing
Auth/DB	Supabase	Authentication, storage
AI Service	Edge Functions	Generate content + LaTeX
Compiler	pdflatex	PDF generation

B. Data Flow

The main data flow of the AI Powered Research Paper Generator consists of five steps. In Step 1, the user authenticates through Supabase Auth, which provides a JSON Web Token (JWT) stored in the client side. In Step 2, the authenticated user submits a paper configuration form that includes paper format (IEEE or APA), title, authors, keywords, number of pages, and additional information. In Step 3, the frontend invokes the generate-paper edge function using supabase.functions.invoke(), sending the paper configuration payload and the user's JWT to validate the authorization. In Step 4, the edge function creates a structured prompt containing the user's parameters and sends it to the LLM API. The API returns a JSON object consisting of two fields: content (the text narrative of the paper) and latex_source (LaTeX source for a complete compiled paper). Both outputs are sent back to the client and are inserted to the research_papers table in PostgreSQL. In Step 5, the client visits the LaTeX editor page with the LaTeX source payload passed via React Router location.state, where the user reviews and edits the generated LaTeX document.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

C. *Database Schema*

Supabase PostgreSQL database maintains two tables. The profiles table stores the metadata of the user—full_name and email—and is linked to the auth.users table through the id column using a foreign key; the table is populated automatically during the signup flow. The research_papers table maintains all the generated paper records, storing the following attributes: user_id (foreign key), title, authors (JSONB array), paper_type (enum IEEE or APA), keywords (text array), page_count (integer), content (text), latex_source (text), additional_info (text), and created_at (timestamp). Row-Level Security (RLS) policies on both tables guarantee that users have read and write permissions exclusively to their own records.

IV. FRONTEND DESIGN

A. *Technology Stack*

The frontend is built as a React 18 single-page application and is bootstrapped with Vite to offer fast build times and hot module replacement during development. TypeScript is used throughout the entire project to enforce type safety, reduce runtime bugs, and improve developer experience with enhanced IDE support. The routing system is provided by React Router DOM v6 to enable declarative and component-based navigation across four primary routes. Customizable UI components are imported from the shadcn/ui library, which is composed of headless, accessible Radix UI primitives styled using Tailwind CSS utility classes. A custom Tailwind theme is defined to add three custom academic color tokens (academic-blue, academic-cream, academic-light).

B. *Routing and Page Structure*

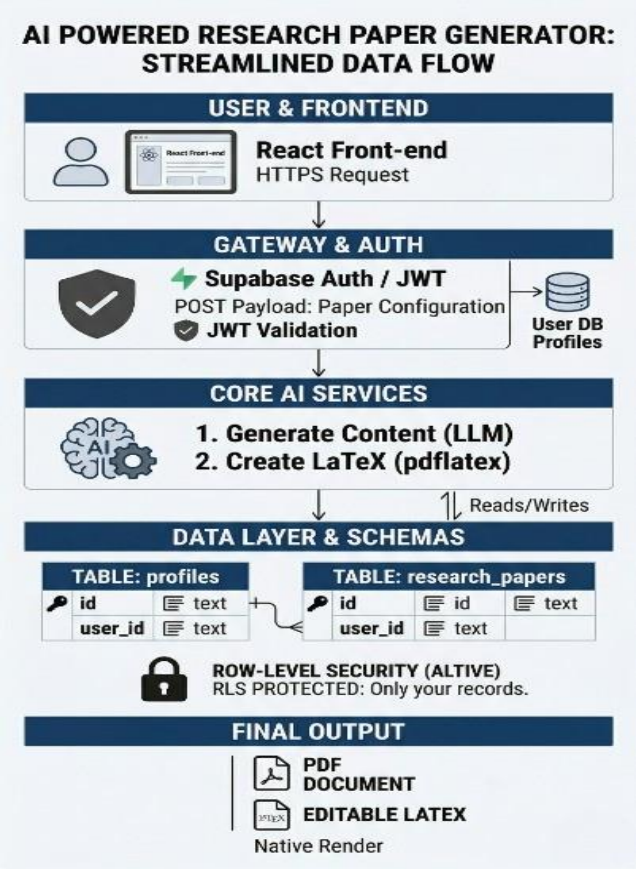
Four main routes are defined in the application: the root route (/) renders the Index page, a marketing landing page with animated UI and feature cards that illustrate the capabilities of the platform. The /auth route renders the Auth page, which contains toggled login and signup forms validated using Zod schema validation library. The /generator route renders the Generator page, the main page for configuring the paper. The /latex-editor route renders the LaTeX editor page, which can only be visited if LaTeX source is passed via React Router location.state payload; otherwise, an IEEE template will be loaded. Route authentication guards are implemented in the Generator component, which on mounting queries supabase.auth.getSession() and redirects unauthenticated users to the /auth route. This approach avoids the complexity of implementing a global authentication context while providing sufficient protection for core functionality.

C. *Paper Configuration Form*

The Generator page displays a form capturing all the necessary parameters to configure the paper. The form supports two paper formats (IEEE and APA), which can be selected using a radio group. The author management subsystem allows users to add and delete author entries, each of which consists of four fields: name, email, affiliation institution, and institutional address. This dynamic list is controlled by the React useState hook with array manipulation methods. All form fields are validated using Zod schemas before submitting the paper generation request. The required fields are paper title, at least one author, and keywords. Optional fields are number of pages and additional information. Once the validation is successful, the form data will be serialized into a JSON payload and then sent to the generate-paper edge function. A loading status indicator will be shown during the asynchronous operation, and error notifications will be delivered via the use-toast hook.

D. *Component Architecture*

Custom reusable UI components are maintained inside the /components/ui directory, adhering to the same structure of shadcn/ui. Examples include Button, Card, Input, Label, Select, RadioGroup, and Textarea, among others. The /hooks directory includes the use-toast.ts hook, which provides a unified notification API consumed across all pages. The /integrations/supabase directory holds the Supabase client singleton instance and the TypeScript type definitions auto- generated from the database schema, ensuring type-safe database operations in the frontend.



V. BACKEND ARCHITECTURE

A. *Supabase Backend*

Supabase is an open-source Firebase alternative based on PostgreSQL, providing authentication, real-time subscriptions,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

storage, and serverless edge functions. In AI Powered Research Paper Generator, Supabase serves as the backend layer, eliminating the necessity of managing a separate server infrastructure. Supabase is initialized with a public API URL and an anon key, while the access control is handled through JWT verification and Row-Level Security policies in the database. Supabase Auth offers email/password authentication out of the box. The sign-in with password and sign-up methods are called directly from the client, with the Auth service managing token issuing, refreshing, and invalidation. The signup flow extends the default behavior by inserting a corresponding entry into the profiles table via supabase.from('profiles').insert() immediately after registration.

B. Generate-paper Edge Function

The generate-paper edge function is the main integration point with AI capabilities. It is built as a Deno-based function running on the Supabase edge function runtime, which provides a globally distributed low-latency execution environment. The edge function is triggered by an HTTP POST request containing the paper configuration payload and Bearer token for authentication purposes. When invoked, the edge function creates a structured system prompt and user prompt targeted to the configured LLM. The system prompt instructs the model to generate an academically rigorous paper conforming strictly to the specified format (IEEE or APA), including all the required structural elements, such as abstract, keywords, numbered sections, in-text citations, and a reference list. The user prompt supplies the paper's title, authors, keywords, target number of pages, and additional information provided by the user. The model is asked to deliver a JSON response object with precisely two fields: content (the full text narrative of the paper) and latex_source (a complete LaTeX document). The LaTeX source includes the appropriate document class (IEEEtran for IEEE, article with natbib for APA), formatted author block, section environments, and bibliography. The edge function processes this JSON response and sends both fields back to the client.

C. Prompt Engineering

To get reliable structured output from LLMs, prompt design is essential. The generate-paper function implements a multi-constraint prompting technique. The system prompt describes: (1) the exact LaTeX document class and packages needed; (2) mandatory structural sections for the target format; (3) the expected length in pages and estimated word count; (4) the required JSON response schema; and (5) formatting constraints, such as double-column layout for IEEE and author-year citations for APA. An important challenge in this prompt design is ensuring that the LaTeX output is syntactically correct and will compile without errors. The system prompt instructs the model to prevent common LaTeX mistakes, like unmatched braces, undefined commands, and missing package imports. Nevertheless, even with these constraints, prompt-level guarantees of compellability cannot be ensured, hence the rationale behind the inclusion of client-side LaTeX validation prior to calling the compile-latex function, as explained in Section VI.

D. Data Persistence

Upon receiving the edge function response, the client saves the generated paper in the research_papers table. This insertion includes all configuration parameters along with the generated content and LaTeX source, allowing users to retrieve and edit previously generated papers. The ability to store LaTeX source in the database is particularly valuable, as it enables the system to act as a lightweight version-controlled repository for the user's generated papers, with a potential future iteration to explicitly version control generated papers.

VI. LATEX COMPILATION

A. Editor Interface

The LaTeX editor page offers a developer-like editing experience for the generated LaTeX source code. It uses the reactsimple-code-editor library, which renders a textarea decorated with syntax-highlighted output generated using Prism.js with a LaTeX grammar definition. The interface shows the user the output of the generated LaTeX source immediately, with different colors representing commands, environments, and text content. The LaTeX editor page fetches its initial content from React Router location.state payload, which is supplied by the Generator page upon navigating to it. Otherwise, the page loads a default IEEE template as a fallback, making it possible to access the editor without executing any backend requests.

B. Client-Side Validation

Upon receiving the edge function response, the client stores the generated paper in the research_papers table. This insertion includes all configuration parameters along with the generated content and LaTeX source, allowing users to retrieve and edit previously generated papers. The ability to store LaTeX source in the database is particularly valuable, as it allows the system to function as a lightweight version-controlled repository of the user's generated papers, with potential future iterations adding explicit versioning.

C. Compile-latex Edge Function

The compile-latex edge function accepts the raw LaTeX source as a string in an HTTP POST request and runs pdflatex to compile the source in a serverless edge function environment. The edge function writes the LaTeX source in a temporary file, invokes pdflatex in non-interactive mode (pdflatex interaction=nonstopmode), reads the generated PDF binary, encodes it to a Base64 data URL, and delivers it to the client. The inclusion of pdflatex in a serverless edge function environment implies certain constraints regarding the availability of LaTeX packages, since the runtime environment does not maintain any TeX distribution. Therefore, the edge function provides a curated list of commonly used packages compatible with the IEEEtran document class, such as amsmath, graphicx, hyperref, cite, and algorithm2e. Packages that depend on external resources, such as TikZ with external libraries, are not included in the current implementation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

D. PDF Rendering

The PDF data URL returned by the edge function is rendered using the HTML object element with a fallback iframe, relying on the native PDF rendering capabilities of the browser. No client-side PDF rendering library (such as PDF.js) is used in this process, which reduces the size of the frontend bundle. Users can also download the raw LaTeX source as a .tex file using the Download LaTeX button, which enables compiling and working with the LaTeX source in an external environment.

VII. CHALLENGES

A. LLM Output Reliability

An important challenge in designing AI-powered applications is the inherent randomness of model output. Despite prompt engineering techniques, sometimes the output might contain syntactical issues that make it impossible to compile LaTeX. While the current design mitigates this issue by validating the structure of the LaTeX source, it does not ensure the output will always compile successfully. A better approach to handle unreliable outputs would be implementing a server-side attempt to compile the LaTeX document and, in case of failure, reprompt the model for a correction.

B. LaTeX Package Availability

Compiling LaTeX server-side with pdflatex in a serverless edge function environment poses restrictions on the available package ecosystem. Certain complex papers might require mathematical notation, special bibliography styles, or advanced formatting options (such as multi-page tables or code listings via listings package) that are not supported by the current LaTeX package selection. Extending the supported package list would require a more persistent environment or a separate LaTeX compilation microservice.

C. Academic Integrity

The use of AI-generated content to craft academic papers raises issues concerning academic integrity. Although the system can be used only as a scaffolding and drafting tool and not as a substitution for original research, the generated content is still generic, since the LLM has no access to domain-specific data, experimental results, and unpublished findings. Therefore, users must revise and supplement the generated draft with their research. At present, the system does not provide any means of plagiarism detection and similarity checks, although it is one of the future priorities for development.

D. Scalability

The current implementation of the system relies on the free tier of Supabase, which imposes constraints on edge function execution time, database storage, and monthly active users. In addition, LLM API calls cause latency ranging from a few seconds to several tens of seconds depending on the length of the document, which may negatively affect the user experience. Furthermore, server-side LaTeX compilation is computationally heavy, and multiple compilation requests concurrently can saturate the edge function's CPU budget.

E. Security

As discussed above, the Supabase anon key is included in the client-side JavaScript bundle. While this is consistent with the security model adopted by Supabase, where the anon key is public, with data protection secured by RLS policies at the database level, the API key of the LLM provider must be kept safe as a Supabase Edge Function secret, never to be exposed to the client. Misconfiguration of this secret would result in the API key leak and unauthorized usage fees.

VIII. FUTURE WORK

The following improvements are planned for further development of AI Powered Research Paper Generator:

- Citation and Reference Management: Integration with academic databases (CrossRef, Semantic Scholar, or Google Scholar APIs) to generate verified live citations, as opposed to the synthesized references by the LLM that may contain incorrect DOI numbers or publication details.
- Plagiarism and Originality Detection: Integration with a similarity detection service (Turnitin API or an open-source equivalent) to assess the originality score of the generated draft.
- Multi-Format Export: Apart from exporting the document in PDF format through LaTeX compilation, generate DOCX format output using Pandoc.
- Iterative Paper Refinement: Chat-based interface for refining specific parts of the generated document using natural language instructions.
- Template Library: Preconfigured templates for various IEEE and ACM conferences or journal formats, which incorporate exact formatting guidelines.
- Compilation Robustness: Implementing a self-healing compilation process inside the generate-paper edge function, where compilation errors are provided back to the LLM and corrected.
- User Collaboration: Supporting multiuser paper editing and version control, as well as paper comments and discussions to facilitate research collaboration.

IX. CONCLUSION

In this study, AI Powered Research Paper Generator—a proposed web-based platform for automated generation of IEEE- and APA-formatted academic papers using LLMs—was presented. The system utilizes React 18 TypeScript frontend and Supabase Backend-as-a-Service platform to offer serverless edge functions for AI-powered content generation and server-side LaTeX compilation. The architectural design provides an end-to-end workflow from user authentication and configuration to downloading a formatted PDF file. This system aims to solve a real problem in academia, especially

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

among engineering students and early researchers, by minimizing the difficulty of creating structurally compliant draft papers. Simultaneously, it recognizes the limitations of the platform as a scaffolding tool that requires substantial human involvement, domain expertise, and original research to produce publication-worthy material. The architecture is easily extensible, and the system is scalable, with clear paths toward citation verification, plagiarism detection, iterative paper refinement, and multi-format exports. Currently, its frontend and database schema are complete, and edge functions are also developed. Further work will focus on improving the compilation process, increasing output reliability with iterative prompting techniques, and expanding LaTeX package support. AI Powered Research Paper Generator demonstrates the practical feasibility of integrating modern serverless architectures with LLM capabilities to build powerful productivity tools for academic writing workflows.

REFERENCES

[1] T. B. Brown et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.

[2] OpenAI, "GPT-4 Technical Report," arXiv preprint arXiv:2303.08774, 2023.

[3] K. Guo, S. Shi, and X. Wang, "Automatic Academic Paper Abstract Generation Using Transformer-Based Models," in Proc. IEEE Int. Conf. on Natural Language Processing and Knowledge Engineering (NLP-KE), 2021, pp. 1–6.

[4] C. Kreutz and R. Schenkel, "Scientific Paper Recommendation Systems: A Literature Review of Recent Publications," arXiv preprint arXiv:2201.00682, 2022.

[5] E. Grundkiewicz, M. Junczys-Dowmunt, and K. Heafield, "Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation," in Proc. NAACL-HLT, 2019, pp. 2590–2600.

[6] J. Hammersley, "Overleaf: Authoring, Collaboration and Publishing Tools for Scientific and Academic Writing," Learned Publishing, vol. 30, no. 2, pp. 175–180, 2017.

[7] Y. Zhang et al., "Server-Side Document Compilation in Cloud Environments: A Performance Analysis," in Proc. IEEE Int. Conf. on Cloud Computing (CLOUD), 2020, pp. 452–460.

[8] Supabase Inc., "Supabase Edge Functions Documentation," 2023. [Online]. Available: https://supabase.com/docs/guides/functions

[9] M. Abad et al., "Serverless NLP Inference Pipelines: Trade-offs Between Cold Start Latency and Cost," in Proc. ACM Symposium on Cloud Computing (SoCC), 2022, pp. 314–327.

[10] D. Knuth, "The TeXbook," Addison-Wesley Professional, 1986.

[11] M. Goossens, F. Mittelbach, and A. Samarin, "The LaTeX Companion," Addison-Wesley, 1994.