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

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

FlashDoc Pro: A Secure, Multi-Engine AI Pipeline for Automated Mathematical and Diagrammatic Typesetting

Lokesha B Acharya
Associate Professor
Deparment of Electrical & Electronics Engineering
Srinivas Institute of Technology
Mangaluru,Karnataka,India

Sudhir P
Assistant Professor
Department of Electrical & Electronics Engineering
Srinivas Institute of Technology
Mangaluru, Karnataka, India

Abstract **- The digitization of complex mathematical formulations and technical diagrams—particularly within advanced fields such as electrical engineering and quantum computing—remains a significant bottleneck in academic research and technical documentation. Traditional Optical Character Recognition (OCR) systems frequently fail to parse dense matrices, sub-scripted variables, and multi-line derivations. While modern Vision-Language Models (VLMs) offer superior recognition, their web-based interfaces necessitate highly manual, repetitive copy-paste workflows that disrupt the research process. This paper introduces FlashDoc Pro, a standalone desktop architecture designed to bridge the gap between raw visual data capture and automated academic typesetting. Operating as a robust wrapper for multi-provider VLMs, the system integrates a local cryptographic security vault using the Windows Data Protection API (DPAPI), a memory-safe image ingestion gallery, and a highly resilient exponential backoff engine for API traffic management. By automatically routing visual data through selected AI engines (such as Google Gemini and OpenAI) and directly compiling the output via Pandoc, FlashDoc Pro successfully converts screen-captured equations into perfectly formatted, editable LaTeX and Microsoft Word documents with near-zero latency, significantly streamlining the technical documentation pipeline.**

Keywords **-** ***Vision-Language Models, LaTeX, Optical Character Recognition, AI engines, Maximum Power Point Tracking.***

I. INTRODUCTION

The formulation of advanced academic research frequently involves the manipulation of dense mathematical models. For instance, developing algorithms for Maximum Power Point Tracking (MPPT) in photovoltaic panels or simulating quantum computing logic gates requires extensive mathematical notation. Transitioning these derivations from whiteboards, handwritten notes, or legacy PDF formats into editable digital documents is traditionally a manual, error-prone, and time-consuming task.

While standard Optical Character Recognition (OCR) technology has matured for natural language, it fundamentally lacks the spatial reasoning required to accurately digitize multi-line equations or complex electrical circuit matrices. Recently, Vision-Language Models (VLMs) have demonstrated extraordinary capability in image-to-LaTeX translation. However, utilizing these models typically requires interacting with web-based chat interfaces, manually prompting the AI to ignore conversational filler, and manually copying the resulting code into a compiler.

This paper proposes FlashDoc Pro: a seamless, offline-first user interface that automates the entire prompt engineering, visual processing, and compilation pipeline. By utilizing native Windows libraries and open-source compilation tools, the application provides a highly resilient, automated pathway from visual capture to formatted document..

II. SYSTEM ARCHITECTURE

A. Security and Authentication

Given the utilization of premium API endpoints, securing user credentials locally is paramount.

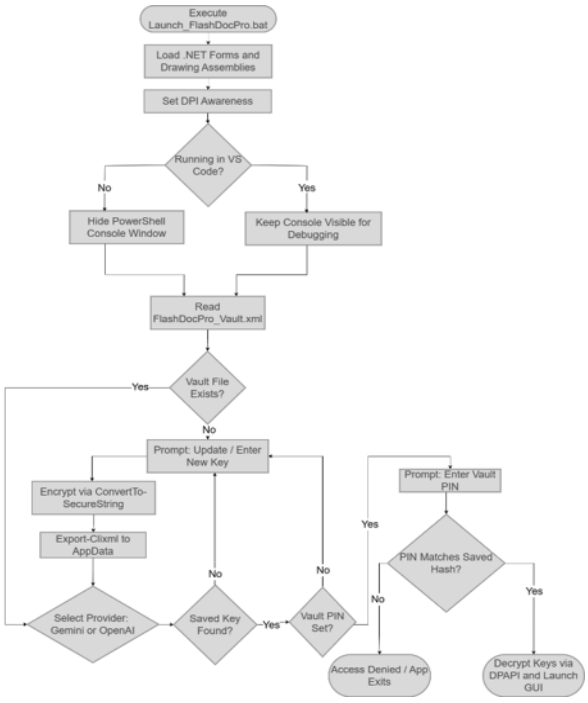

Figure 1: The Security & Initialization Gateway

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

As illustrated in Figure 1: The Security & Initialization Gateway, FlashDoc Pro bypasses plain-text key storage by leveraging the Windows Data Protection API (DPAPI). The application encrypts API keys using the Convert-to-Secure-String protocol, tying the decryption capability to the active user's Windows profile. Furthermore, the architecture implements a secondary application-level Vault PIN challenge. This two-factor local authentication ensures that even if the encrypted FlashDocPro_Vault.xml payload is exfiltrated, the API endpoints cannot be accessed without the user's specific integer matrix

B. Ingestion And User Interface

The application initiates via a stealth-launch mechanism designed to suppress background terminal windows, preventing visual clutter while dynamically detecting IDE environments (such as Visual Studio Code) to preserve debug logs when necessary. The graphical user interface (GUI) is constructed using DPI-aware .NET Windows Forms. To ensure system stability during batch processing, visual inputs—whether acquired via the built-in "Smart Snip" screen capture protocol or manual clipboard pasting—are stored in a memory-safe array rather than being prematurely written to the local disk.

C. Multi-Provider Ai Routing

To prevent vendor lock-in and ensure continuous availability, the system features a multi-provider routing engine, detailed in figure 2.

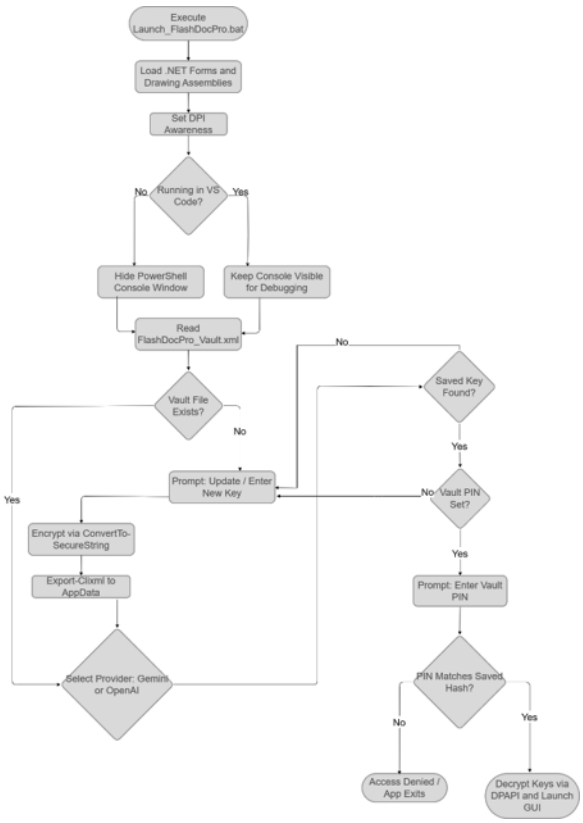

Figure 2: The Core Engine & AI Routing.

. The architecture interfaces with both Google Gemini and OpenAI Vision REST APIs. A critical component of this routing is the automated prompt injection. The system forces the VLM to act strictly as a LaTeX typesetting engine by appending a hidden prompt architecture to every request. This prompt enforces rigorous formatting rules, demands matching bracket delimiters (\left, \right), and explicitly prohibits conversational prose, ensuring the output is pure compilation-ready code.

D. Resilience and Fallback Mechanisms

Interacting with cloud-based AI endpoints introduces the risk of HTTP 429 (Rate Limit) and 503 (Server Unavailable) errors, particularly when processing large batches of dense technical diagrams. FlashDoc Pro mitigates this through a localized try/catch exponential backoff loop. If an API request fails, the system delays the retry attempt by an exponentially increasing margin (e.g., 1s, 2s, 4s, 8s). Crucially, if the primary reasoning model continuously fails, the architecture implements a "Global Survivor" fallback strategy, automatically shifting network traffic to a highly available, lightweight model (such as Gemini Flash) to guarantee batch completion without manual user intervention.

E. Compilation Pipeline

Upon successful extraction of the LaTeX strings from the visual data, the text buffer is sanitized to remove markdown artifacts.

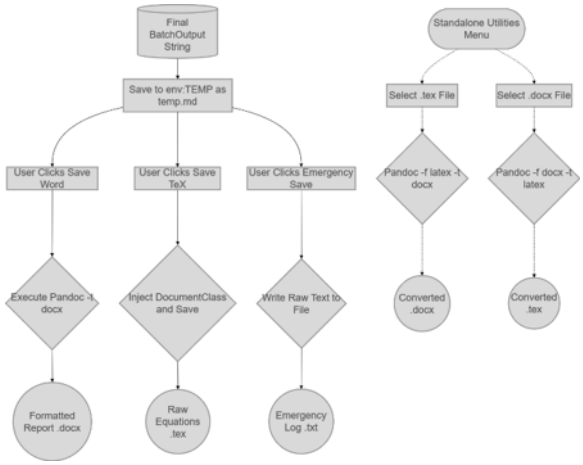

Figure 3: The Compilation & Output Pipeline,

As shown in Figure 3: The Compilation & Output Pipeline, this raw data is then channeled directly into a localized Pandoc execution engine. The pipeline concurrently generates standard .tex files for raw mathematical manipulation and fully rendered .docx academic reports, allowing for immediate integration into standard word processors.

.

III. IMPLEMENTATION METHODOLOGY

The system is constructed entirely using PowerShell 5.1+ and the .NET Framework 4.5+ (specifically leveraging System.Windows.Forms and System.Drawing). This deployment strategy was selected to ensure native execution on Windows operating systems without requiring users to install heavy secondary dependencies such as Python environments or Node.js packages. Application state management is achieved by maintaining session data in volatile memory arrays, utilizing non-blocking UI thread updates ([Application]::DoEvents()) to prevent interface freezing during high-latency API calls. Furthermore, an emergency save protocol is embedded within

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

the compilation pipeline to dump the raw text buffer to the local disk in the event of an unexpected termination, preventing data loss during extensive batch processes.

IV. PERFORMANCE AND EVALUATION

A. Latency vs. Accuracy Trade-offs

Real-world deployment of the tool revealed a distinct operational trade-off between API latency and analytical precision. Utilizing a lightweight model (e.g., Gemini 2.5 Flash) resulted in highly "snappy" responses with low network latency, making it the optimal daily driver for standard circuit diagrams and linear equations. Conversely, processing highly convoluted, multi-line quantum computing derivations or deeply nested MPPT matrices benefited from the deployment of heavier reasoning models (e.g., Gemini 3.0 Flash or 1.5 Pro). While these models introduced slight processing latency due to increased server-side computation, the resulting architectural accuracy of the LaTeX significantly reduced the need for manual post-compilation editing.

B. Compilation Reliability

A secondary evaluation of the output data indicated that enforcing simplified LaTeX delimiters (utilizing standard $$ equation blocks and \begin{aligned} environments) dramatically improved the success rate of the Pandoc conversion pipeline. Preventing the AI from generating highly customized or esoteric LaTeX packages ensured that the resulting .docx files rendered cleanly natively within Microsoft Word.

V. CONCLUSION AND FUTURE WORK

FlashDoc Pro demonstrates that combining local cryptographic security, resilient exponential backoff algorithms, and multi-provider VLM routing within a native desktop interface drastically reduces the friction of academic typesetting. By automating the prompt engineering and compilation phases, researchers can digitize complex technical documentation seamlessly. Future iterations of this architecture will explore the integration of local Small Language Models (SLMs) via frameworks such as Ollama. Operating inference entirely on-device would remove cloud API dependencies, eliminate rate limits and further secure proprietary research data. Additionally, porting the core logic to cross-platform frameworks could extend this robust typesetting pipeline to macOS and Linux environments

REFERENCES

[1] Google Cloud, "Gemini API Documentation," Google for Developers, 2026. [Online].
[2] OpenAI, "Vision and Text Generation API Reference," OpenAI Documentation, 2026. [Online].
[3] J. MacFarlane, "Pandoc: A Universal Document Converter," Pandoc User's Guide, 2026. [Online].
[4] Microsoft Corporation, "Windows Data Protection (DPAPI)," Windows Win32 API Documentation, 2026. [Online].
[5] Microsoft Corporation, "System.Windows.Forms Namespace," .NET API Browser, 2026. [Online].