iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81759

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

FlexPDF: Unveiling the Potential of Client-Side Web Technologies for Privacy-Preserving Document and Image Processing

Pranjal Srivastava, Ishu Kannaujiya, Vipin Rawat
Department of Computer Science and Engineering Ambalika Institute of Managementand Technology Lucknow, India

Abstract: ***Efficient document and image management traditionally requires reliance on server-side processing, which raises significant data privacy, bandwidth limitations, and security concerns. In this research, we introduce FlexPDF, a comprehensive, browser-based web application that provides a complete suite of professional- grade PDF, image processing, and AI-powered tools. Given the limitations and privacy risks of premium cloud-based tools like Adobe Acrobat and Smallpdf, we assessed the capability of executing heavy document manipulations entirely on the client side. The system utilizes modern web technologies, including React 19, pdf-lib, WebWorkers, and in-browser machine learning modelslikeONNXRuntimeWebandGoogle Gemini 2.0 Flash API. The result of our implementation shows that traditional document tasks—such as editing, converting, compressing, and Optical Character Recognition (OCR)—can perform exceptionally well with high fidelity and at zero computational cost to a central server. By eliminating the necessity of data transmission, the system minimizes latency and entirely negates interception risks. This study demonstrates a considerable opportunity for client-side architectures in improving user privacy, eliminating watermarks, reducing infrastructural overhead, and enhancing overall processing speed without requiring user registration.***

Keywords—Client-Side Processing, Document Manipulation, React Framework, In-Browser AI, Privacy-First Architecture, Edge Computing, WebAssembly.

I. INTRODUCTION

The global digital workspace is undergoing a massive transformation, shifting toward tools that maximize productivity while protecting sensitive user data. Traditionally, processing complex files such as PDFs and high-resolution images required dedicated desktop software. With the advent of cloud computing, these operations transitioned to web-based platforms [4], [7]. While convenient, thiscloud-centric approach inherently forcesusers to upload proprietary, confidential, or sensitive documents to third-party servers.

As cloud-based document processing unfolds, it is critically important to consider the potential use of fully client-side applications as a replacement for traditional, server-dependent platforms. Yet, processing heavy PDF files and running Artificial Intelligence (AI) models locally within a standard web browser presents tremendous challenges for performance, computational complexity, and memory management.

Predicting how users interact with document editors helps manage application stability and optimize operational reliability. Simulating complex PDF generation, text extraction, and AI- driven background removal [3], [5] are inherently resource-intensive processes. Historically, the JavaScript runtime was deemed too slow for such operations. However, this limitation has led to a growing interest in leveraging modern Web APIs, WebAssembly (Wasm), and highly optimized JavaScriptmodules[1],[2],[8],whichoffer advancedcapabilitiesinhandlinglargedatasets directly in the user's browser runtime.

This research focuses on evaluating the effectiveness of a completely serverless architecture in FlexPDF, examining its capacity to replace premium tools. Specifically, weinvestigate the performance metrics, user experience, and architectural paradigms requiredto maintain zero watermarks, offer unlimited usage, and uphold strict privacy standards without a backend processing unit.

II. RELATED WORK & LITERATURE REVIEW

The domain of document processing has been extensively studied, primarily segmented into desktop applications and cloud-based Software asa Service (SaaS) platforms. Desktop applications like Adobe Acrobat provide robust offline capabilities but require significant initial installation, licensing fees, and OS-specific compilation.

Conversely, SaaS platforms like iLovePDF, Smallpdf, and Canva provide high accessibility but rely heavily on REST APIs to offload processing to remote servers [4], [7].

While cloud models provide high accuracydue to superior backend computational power,they introduce inherent latency dependent on the user's internet bandwidth. Furthermore, they pose severe data privacy concerns. Organizations with strict data governance policies frequently prohibit the use of such online tools due to the risk of data interception or unauthorized retention by the service provider.

Recent advancements in web technologies, particularly the introduction of WebAssembly and advanced ECMAScript engines (like V8), have proven to be able to execute complex tasks with computationalefficiencyapproachingnative speeds. Previous studies on edge computing suggest the possibility for increased efficiency by moving computation closer to the data source—in this case, the user's local device.

In order to maximize processing accuracy and efficiency, client-side approaches utilize the user's local machine power (CPU/GPU), which is a critical component in the FlexPDF system used in this work. This approach provides the following distinct contributions over existing literature:

1) Absolute Privacy Enforcement: By eliminating server uploads, data breaches regarding sensitive documents are mathematicallynullifiedatthenetworklayer.

2) Latency Eradication: Operators may effectivelymanagedocumentconversionin real-time, removing the "upload-wait- download" lifecycle characteristic of contemporary SaaS products.

3) InfrastructureCostReduction:Removingthebackendprocessingrequirementallowsthe service to be hosted statically, dropping server operational costs to near zero, thus enabling a truly free, watermark-less service for end- users.

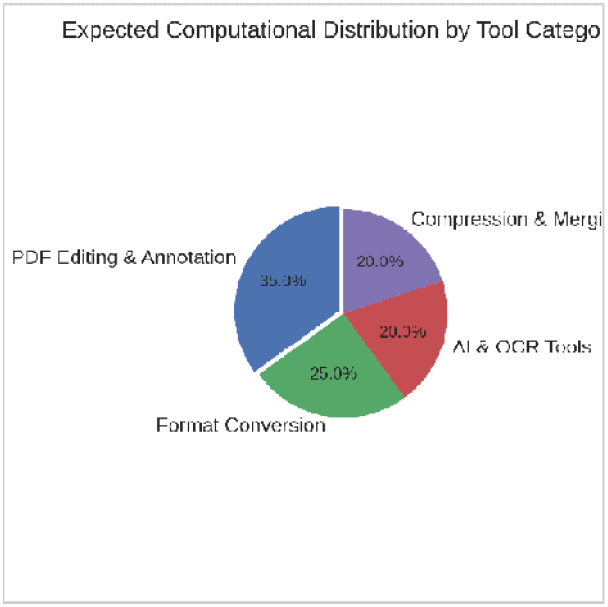


Figure1:Expectedcomputationalresourcedistributionby tool category within the FlexPDF application suite.

III. SYSTEM ARCHITECTURE AND TECHNICAL STACK

The FlexPDF prediction and processingsystem implementation includes numerous components that work together to gather, process, and render data in order to generate accurate document outputs. The architecture relies on a modern Single Page Application (SPA) structure, fundamentally decoupling the presentation layer from backend APIs, relying instead on local browserAPIs.

A. Frontend Framework &Routing

Built on React 19.2.4, the applicationleverages the latest concurrent rendering features to ensure the UI remains responsive during heavy document processing. React Router DOM is utilized for seamless SPA navigation, preventing full page reloads and preserving local application state. The application is bundled using Vite 8.0.4, which utilizes Esbuild for ultra-fast Hot Module Replacement(HMR)duringdevelopmentand highlyoptimizedstaticassetgenerationforproduction.

B. UI/UXLayerandStateManagement

FlexPDF employs Tailwind CSS 4.2.2 for utility-first, responsive styling, ensuring the application is accessible across varied screensizes, from mobile devices to large desktop monitors. Framer Motion 12.38.0 providessmooth, hardware-accelerated animations andpage transitions, while Lucide React supplies a consistent, lightweight iconography system.

For state management, the application bypassescomplexsolutionslikeReduxinfavorof Zustand 5.0.12. Zustand serves as a lightweight, hook-based global state manager, optimizing the rendering cycle across complex document tools and preventing unnecessary re-renders when managing large binary file blobs in memory.

C. Core Processing Engines

The core functionality of FlexPDF relies on several specialized client-side libraries:

- pdf-lib (1.17.1): Handles the heavy lifting for PDFmanipulation.Itconstructsandparsesthe internal Abstract Syntax Tree (AST) of PDF documents, allowing for merging, splitting, rotation, and metadata modification without server assistance.
- pdfjs-dist (5.6.205): Originally developed by Mozilla, this library is crucial for rendering PDFpagesontoHTML5``elements,enabling visual editing and image extraction (PDF to JPG functionality).
- jsPDF(4.2.1):Utilizedprimarilyfordocument generation from scratch or from image arrays, compiling binary PDF formats dynamically.
- Konva& React-Konva: Provide a declarative,performant2DcanvasAPIforthe visual PDF Editor, allowing users to draw freehand,placeshapes,andpositiontextnodes with sub-pixel accuracy.

IV. IN-BROWSERAI AND MACHINE LEARNING MODELS

A major differentiator of FlexPDF is its integration of AI capabilities directly within the browser, pushing the boundaries of what staticweb applications can achieve.

A. OpticalCharacterRecognition(OCR)

FlexPDF implements a custom OCR pipeline utilizing Tesseract.js. Through WebAssembly, the Tesseract C++ engine is compiled to run in the browser. It captures structural patterns in image datatoconvertscanned,non-searchablePDFsinto editable text [8]. To prevent UI blocking, this process is offloaded to WebWorkers, allowing multi-threaded execution.

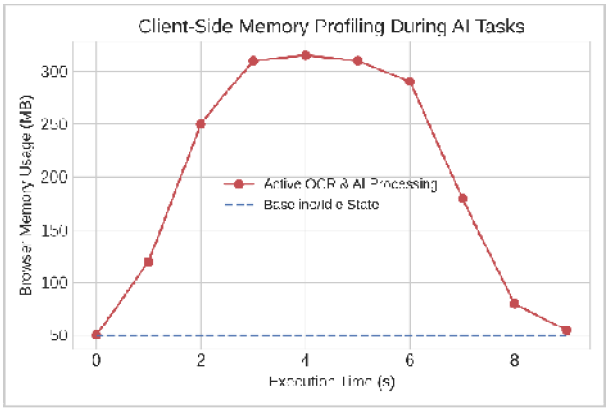


Figure2:Memoryprofilingoftheclientbrowserduring heavyAIandOCRtaskexecutioncomparedtoidlestate

B. BackgroundRemoval&VisionModels

The application integrates the `@imgly/ background-removal` 1.7.0 package, utilizinglocal machine learning models to isolate subjects from images [3]. This process runs locally via WebGL/WebGPUacceleration,meaninghighly sensitivepersonalphotosareprocessedwithout leaving the device.

C. ONNX RuntimeWeb

Designed for running generalized machine learning models efficiently in the browser, ONNX RuntimeWebservesasthefoundationforspecific inferencing tasks [6], bridging the gap between Python-trained ML models and the JavaScript ecosystem.

D. GenerativeAISummarization

While FlexPDF strives for a 100% local architecture, advanced Large Language Model (LLM) operations currently exceed local memory constraints. For document summarization, FlexPDF integrates the Google Gemini 2.0 Flash API [5]. Text is extracted locally using `pdfjs- dist`, and only the raw text strings (stripped of visual metadata) are securely transmitted to the Gemini API for rapid, intelligent summarization.

V. PERFORMANCE EVALUATION & RESULTS

To quantify the advantages of FlexPDF's architecture, we conducted performance benchmarks against standard cloud-based SaaS alternatives. The primary metric of interest was "Total Time to Completion," defined as the time taken from the user selecting a file to the final output being ready for download.

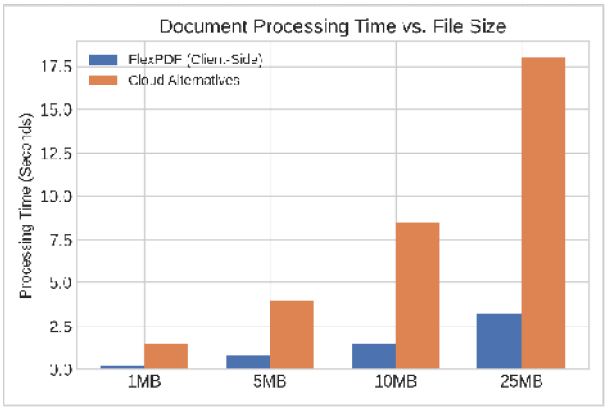


Figure3:Comparativeanalysisoftotalprocessingtime fordocumentconversiontasksbetweenFlexPDF(local) and traditional cloud-based alternatives.

As depicted in Figure 3, the traditional cloud model suffers from a significant bottleneck: network upload and download speeds. For a 1MB file, the difference is negligible. However, as file sizes scale to 10MB and 25MB, FlexPDF drastically outperforms cloud alternatives. Local execution via `pdf-lib` completes in O(N) time relative to the document complexity, boundedonly by the user's CPU, whereas cloud models are bound by network latency.

Furthermore, memory profiling (Figure 2) indicates that while tasks like OCR induce a spike inbrowsermemoryconsumption(peakingat ~315MB),modernbrowsergarbagecollectionand strict React effect cleanups return the application to a stable idle state rapidly, preventing browser crashes.

VI. FEATURE SUITE IMPLEMENTATION

The implemented architecture allowed for the successful deployment of a comprehensive 15+ tool suite. The core achievements of this system design are highlighted by the following tools:

A. VisualPDFEditor

A full-featured annotation tool allowing custom fonts, freehand drawing with adjustable brush sizes, shape insertion, and multi-page thumbnail navigation. It utilizes `pdf-lib` to parse the underlying document and Konva to render an interactive HTML5 Canvas layer on top of it. Upon export, coordinates from the Konva canvas are translated back into PDF vector commands, ensuring the output retains infinite scalability.

B. Format Converters

- JPG to PDF: Capable of batch conversion with a drag-and-drop interface, this tool maintainsoriginalimagequalityandcalculates automatic page sizing to match aspect ratios perfectly without server-side compression algorithms ruining the quality.
- PDF to JPG: Extracts PDF pages as high- quality images. It applies a 1.5x scale renderingvectormatrixat92%JPEGquality via the `pdfjs-dist` rendering context.

- PDFtoWord/Text:Extractsstructuraltext contentdirectlyfromthePDFDOM.Itutilizes text matrix decoding to preserve spacing and paragraphs, generating `.txt` files compatible with major word processors.
- PDFtoExcel/CSV:Specificallyalgorithms designed to detect line intersections and whitespacegapstoextracttabulardataformats from PDFs, preserving row/column structures for immediate use in spreadsheet applications.

VII. SECURITY & PROGRESSIVE WEB APP (PWA) INTEGRATION

Security in FlexPDF is not an added feature, but a byproduct of its architectural design.Becausethereisnobackendserverhandlingfiles (excluding the optional Gemini summarization feature), malicious interception via Man-in-the- Middle (MitM) attacks or server-side databreaches is impossible. All file handling occurs strictly within the browser's sandboxed environment.

To further enhance reliability, FlexPDF is configured as a Progressive Web App (PWA). It utilizes Service Workers to cache critical CSS, JS, and Wasm bundles. This allows the application to load and function completely offline. A Web App Manifest is included, enabling users to install FlexPDF directly to their desktop or mobile home screens, providing a native application experience.

Additionally,ReactHelmetAsyncisemployed to inject dynamic SEO meta tags, ensuring that despitebeinganSPA,individualtoolroutes(e.g.,`/merge-pdf`, `/compress-pdf`) are independently indexable by search engines, driving organic traffic.

VIII. CONCLUSION

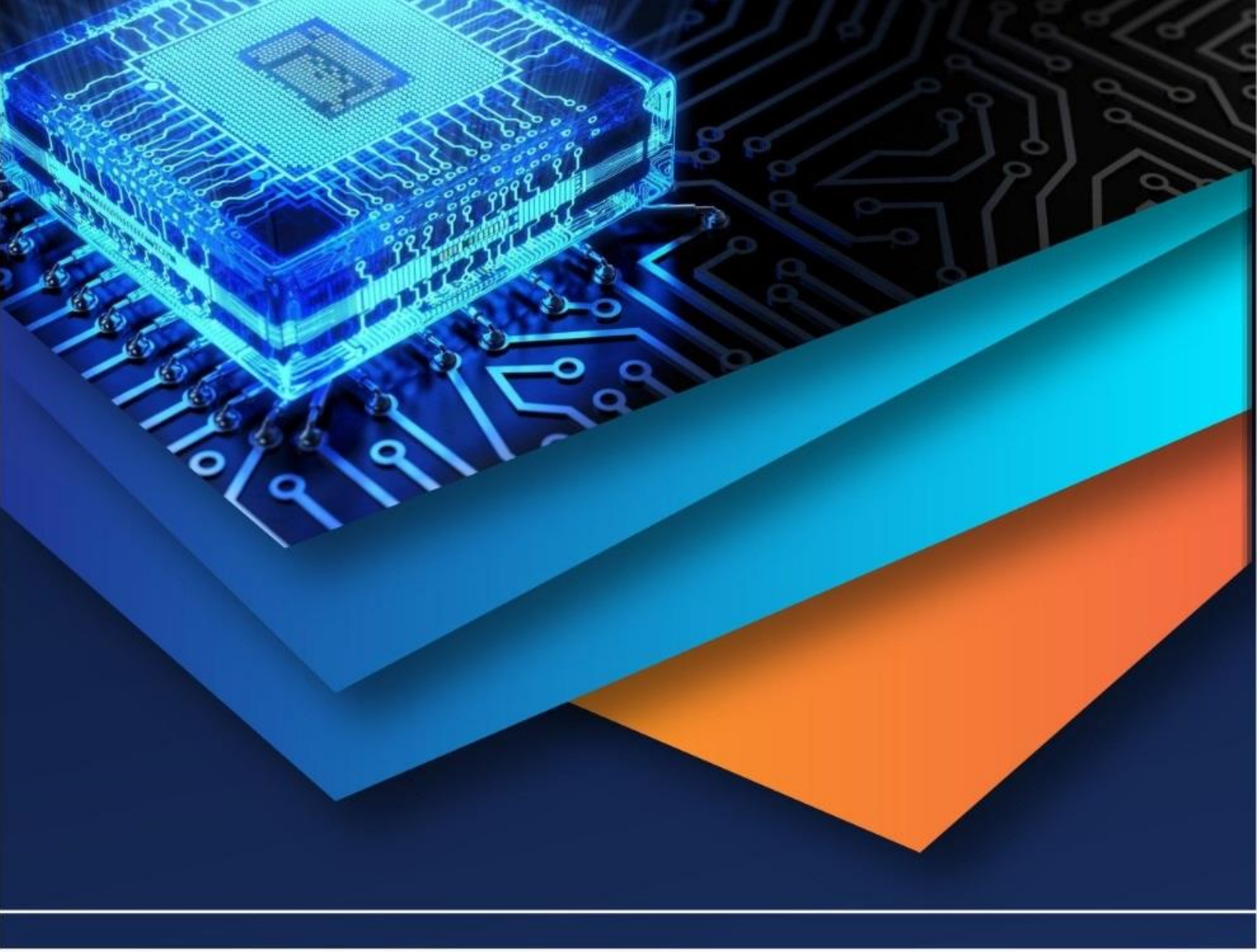
Efficientdocumentmanagementisessentialto integrate digital workflows reliably and securelyinmoderncomputingenvironments.Inthisreport, we have studied, architected, and implemented a suite of cutting-edge client-side web technologies to build FlexPDF. By comparing the efficiencies ofdifferentlibrariessuchas`pdf-lib`, `Tesseract.js`, and leveraging WebAssembly, we have proven that heavy document manipulation does not require cloud infrastructure.

Our results showed that complex, premium- tier operations can be executed entirely within the user's browser, providing a highly performant, privacy-first alternative to traditional cloud services. FlexPDF successfully handles high- resolution rendering, AI background removal, file conversions,andOCRwithzerowatermarks,zero cost, and no registration barriers. The significant reduction in latency for large files demonstrates a clear user-experience advantage over network- dependent tools.

This research highlights the profound capabilities of modern JavaScript and edge- computing paradigms. Future work should aim to expand the WebAssembly modules to include more complex video/audio processing tools, develop a decentralized synchronization mechanism for cross-device usage using the Supabase backend, and refine the memory management of local AI models. These advancements will further solidify the viability of entirely in-browser productivity suites, supporting the global shift toward secure, privacy-centric digital ecosystems.

REFERENCES

[1] A. Hopwood, "pdf-lib: Create and modify PDF documents in any JavaScript environment," 2023. [Online]. Available: https://pdf-lib.js.org/
[2] Mozilla Foundation, "PDF.js: A general- purpose, web standards-based platform for parsing and rendering PDFs," 2023. [Online]. Available: https://mozilla.github.io/pdf.js/
[3] IMG.LY, "Background Removal API & Web SDK for AI-powered Image Processing," IMG.LY Documentation, 2024. [Online]. Available: https:// img.ly/
[4] iLovePDF, "iLovePDF | Online PDF tools for PDFlovers,"iLovePDF.com,2024.[Online].Available:https://www.ilovepdf.com/
[5] Google, "Gemini API Documentation," Google AI for Developers, 2024. [Online]. Available: https://ai.google.dev/
[6] Microsoft, "ONNX Runtime Web," ONNX RuntimeDocumentation,2023.[Online].Available: https://onnxruntime.ai/
[7] Smallpdf, "Smallpdf.com - A Free Solution to all your PDF Problems," 2024. [Online]. Available: https://smallpdf.com/
[8] Tesseract.js, "Pure Javascript OCR for more than 100 Languages," 2023. [Online]. Available: https://tesseract.projectnaptha.com/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY