



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

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 Management and 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 models like ONNX Runtime Web and Google 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, this cloud-centric approach inherently forces users 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 JavaScript modules [1], [2], [8], which offer advanced capabilities in handling large datasets 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, we investigate the performance metrics, user experience, and architectural paradigms required to 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 as a 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 accuracy due 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 computational efficiency approaching native 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 mathematically nullified at the network layer.
- 2) **Latency Eradication:** Operators may effectively manage document conversion in real-time, removing the "upload-wait-download" lifecycle characteristic of contemporary SaaS products.
- 3) **Infrastructure Cost Reduction:** Removing the backend processing requirement allows the service to be hosted statically, dropping server operational costs to near zero, thus enabling a truly free, watermark-less service for end-users.

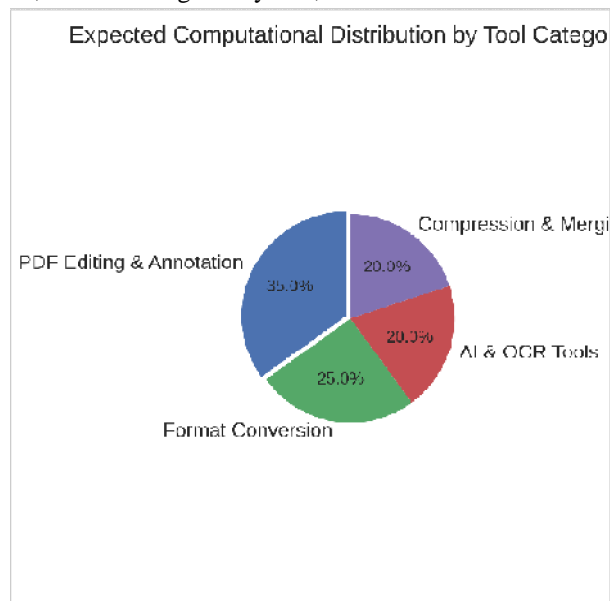


Figure 1: Expected computational resource distribution by tool category within the FlexPDF application suite.

III. SYSTEM ARCHITECTURE AND TECHNICAL STACK

The FlexPDF prediction and processing system 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 browser APIs.

A. Frontend Framework & Routing

Built on React 19.2.4, the application leverages 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) during development and highly optimized static asset generation for production.

B. UI/UX Layer and State Management

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

For state management, the application bypasses complex solutions like Redux in favor of 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 PDF manipulation. It constructs and parses the 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 PDF pages into HTML5 elements, enabling visual editing and image extraction (PDF to JPG functionality).
- jsPDF (4.2.1): Utilized primarily for document generation from scratch or from image arrays, compiling binary PDF formats dynamically.
- Konva & React-Konva: Provide a declarative, performant 2D canvas API for the visual PDF Editor, allowing users to draw freehand, place shapes, and position text nodes with sub-pixel accuracy.

IV. IN-BROWSER AI AND MACHINE LEARNING MODELS

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

A. Optical Character Recognition (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 data to convert scanned, non-searchable PDFs into editable text [8]. To prevent UI blocking, this process is offloaded to WebWorkers, allowing multi-threaded execution.

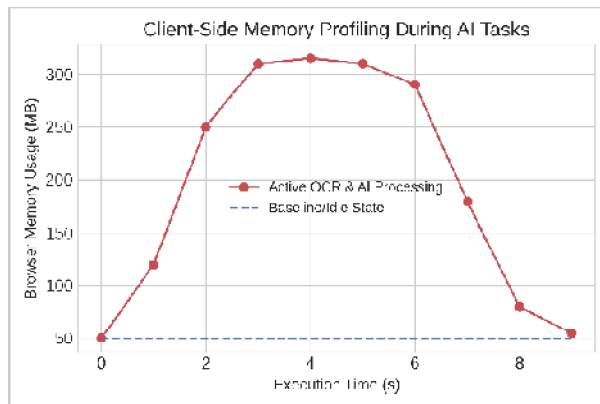


Figure 2: Memory profiling of the client browser during heavy AI and OCR task execution compared to idle state

B. Background Removal & Vision Models

The application integrates the '@imgly/background-removal' 1.7.0 package, utilizing local machine learning models to isolate subjects from images [3]. This process runs locally via WebGL/WebGPU acceleration, meaning highly sensitive personal photos are processed without leaving the device.

C. ONNX Runtime Web

Designed for running generalized machine learning models efficiently in the browser, ONNX Runtime Web serves as the foundation for specific 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.

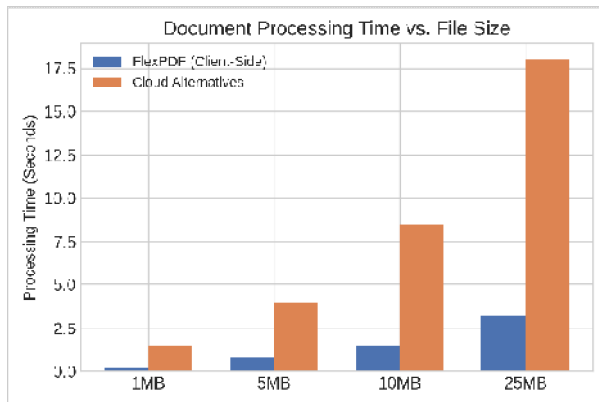


Figure3:ComparativeanalysisoftotalprocessingtimefordocumentconversiontasksbetweenFlexPDF(local) and traditional cloud-based alternatives.

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, bounded only 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 in browser memory consumption (peaking at ~315MB), modern browser garbage collection and 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 maintains original image quality and calculates 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 rendering vector matrix at 92% JPEG quality via the `pdfjs-dist` rendering context.

- PDFtoWord/Text: Extracts structural text content directly from the PDF DOM. It utilizes text matrix decoding to preserve spacing and paragraphs, generating `.txt` files compatible with major word processors.
- PDFtoExcel/CSV: Specifically algorithms designed to detect line intersections and whitespace gaps to extract tabular data formats 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. Because there is no backend server handling files (excluding the optional Gemini summarization feature), malicious interception via Man-in-the-Middle (MitM) attacks or server-side data breaches 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, React Helmet Async is employed to inject dynamic SEO meta tags, ensuring that despite being an SPA, individual tool routes (e.g., `/merge-pdf`, `/compress-pdf`) are independently indexable by search engines, driving organic traffic.

VIII. CONCLUSION

Efficient document management is essential to integrate digital workflows reliably and securely in modern computing environments. In this report, we have studied, architected, and implemented a suite of cutting-edge client-side web technologies to build FlexPDF. By comparing the efficiencies of different libraries such as `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, and OCR with zero watermarks, 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 PDF lovers," 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 Runtime Documentation, 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

Call : 08813907089  (24*7 Support on Whatsapp)