Journal of Computer Science and Frontier Technologies
ISSN Print: 3104-4204
ISSN Online: 3104-4212
CD ISSN: 3104-4220
https://doi.org/10.63313/JCSFT.9084

A Private Web-Based Pipeline for Converting Images and PDF Pages into Editable PowerPoint Shapes

Huaxin Zheng
Wenzhou Polytechnic, WenZhou, China
Email: huaxin179@163.com

How to cite this paper: Zheng, H. (2026). A private web-based pipeline for converting images and PDF pages into editable PowerPoint shapes. Journal of Computer Science and Frontier Technologies, 3(3), 28–36. ISSN Print: 3104-4204; ISSN Online: 3104-4212.
https://doi.org/10.63313/JCSFT.9084
Published: 2026-06-10

Copyright © 2026 by author(s) and Erytis Publishing Limited.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/



Abstract

Rasterized slide images and PDF pages are easy to distribute but difficult to reuse because text, layout objects, borders, and background regions are flattened into pixels. This paper presents a private web-based pipeline that converts images and PDF pages into editable PowerPoint files. The proposed system emphasizes local deployment, backend-centered image analysis, optical character recognition, text-region removal, geometric shape reconstruction, an intermediate Shape JSON representation, and .pptx export. The method first normalizes input pages, detects text regions, estimates background colors, extracts primary layout shapes through color clustering and contour analysis, and then reconstructs editable PowerPoint shapes with fill colors, borders, layer order, and placeholder text fields. A lightweight browser interface supports upload, preview, basic shape correction, and re-export, while computation remains on the local server. The current prototype completes the core conversion loop from raster input to editable PowerPoint output and provides a foundation for improved text removal, manual correction, batch PDF processing, and reusable private templates. The study demonstrates a practical engineering approach for transforming static visual documents into reusable presentation assets without relying on public cloud processing.

Keywords

Document Image Analysis; Editable Presentation; Optical Character Recognition; Shape Reconstruction; PowerPoint Automation; Private Deployment

1. Introduction

Slides, presentation screenshots, and PDF pages are widely used for communication, documentation, and knowledge transfer. However, when a slide is exported as an image or embedded in a PDF, its editable structure is lost. Text becomes pixels, rectangular regions become part of a flattened background, and graphical elements can no longer be selected, moved, recolored, or reused in PowerPoint. This problem

becomes more serious when users need to reuse archived slides, translate existing materials, remove original text, or rebuild a visual layout for a new presentation.

A common workaround is to insert the original image as a full-slide background and manually redraw the layout. This method is labor-intensive and produces limited reusability. Another approach is to rely on full document understanding systems, but such systems may be excessive for private use, may require cloud deployment, and may focus on semantic understanding rather than editable presentation reconstruction. For a private user, the practical requirement is narrower: upload a screenshot or PDF page, remove the original text as cleanly as possible, detect the main visual shapes, and export a .pptx file whose major objects can be edited in PowerPoint.

This paper proposes a private web-based tool for converting images and PDF pages into editable PowerPoint shapes. The system is designed for local or private deployment rather than multi-user cloud operation. Its main objective is not to build a complete online PowerPoint editor, but to provide a reliable backend pipeline for image analysis, text detection, shape reconstruction, and PowerPoint generation. The frontend remains intentionally simple and provides upload, preview, basic correction, and export functions.

The main contributions of this work are as follows. First, it defines an end-to-end conversion pipeline from raster page input to editable .pptx output. Second, it introduces a Shape JSON intermediate representation that separates visual structure from final PowerPoint generation. Third, it combines OCR-based text detection, heuristic fallback detection, background color filling, contour extraction, and shape classification in a modular backend. Fourth, it presents a private deployment architecture that supports local processing, debug outputs, and manual correction without requiring cloud sharing, multi-user accounts, or collaboration features.

2. Problem Definition and Design Requirements

2.1. Input and Output Scope

The input of the system is a raster image or a PDF page. Supported image formats include PNG and JPG. For PDF input, each page is converted into a standard raster image before analysis. The expected output is a PowerPoint file whose slide size matches the aspect ratio of the input page. The output should not merely contain the original screenshot as a background image. Instead, the main visual components should be reconstructed as editable PowerPoint shapes.

The target output includes editable rectangles, rounded rectangles, large color blocks, lines, dashed borders, arrows, simple polygons, and empty text placeholders. The original text is detected and removed from the raster image or excluded from reconstructed shapes. New text can later be added by the user as editable PowerPoint text.

2.2. Functional Requirements

The system is designed around six functional requirements.

1. The system should accept common raster images and PDF pages.
2. The system should detect text regions through OCR when OCR dependencies are available.
3. The system should provide a fallback text-region detector when OCR initialization fails or OCR models are unavailable.
4. The system should detect major layout shapes rather than exporting a single full-page bitmap.
5. The system should generate a Shape JSON file that stores geometry, style, layer order, and editable text placeholders.
6. The system should export a .pptx file in which reconstructed shapes can be selected and modified independently.

2.3. Non-Functional Requirements

The system also follows several non-functional requirements. Privacy is central: documents are processed on a local machine or private server. Robustness is prioritized over visual perfection; the tool should still return a valid output when OCR is unavailable. The user interface should remain simple and should not become a complex web-based presentation editor. Debuggability is also required, because image-to-shape reconstruction may fail on difficult pages. Therefore, the system should keep intermediate outputs such as text-box detection previews, shape-detection previews, repaired images, and processing logs.

3. System Architecture

3.1. Overall Workflow

The overall workflow begins with page input and ends with an editable PowerPoint file. A PNG or JPG image is directly normalized. A PDF page is first rasterized and then normalized to a target processing size. The system analyzes the normalized page through OCR text detection, color clustering, edge detection, and contour extraction. It then removes or fills detected text regions, classifies visible components into shapes, determines layer order, and stores the result in a Shape JSON structure. Finally, the PowerPoint generator converts Shape JSON into editable .pptx objects.

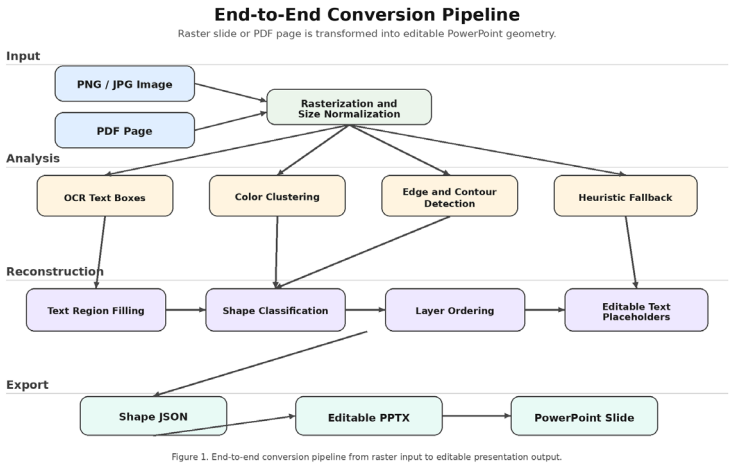


Figure 1. End-to-end conversion pipeline from raster input to editable presentation output.

3.2. Private Deployment Architecture

The proposed tool uses a private deployment architecture. The browser interface provides file upload, original image preview, text-removed preview, detection preview, shape-list display, basic property editing, re-export, and output retrieval. The backend is implemented as a local web server that exposes API endpoints for upload, processing, editing, and export. Local storage is used for uploaded files, debug images, project.json, Shape JSON data, and generated .pptx files.

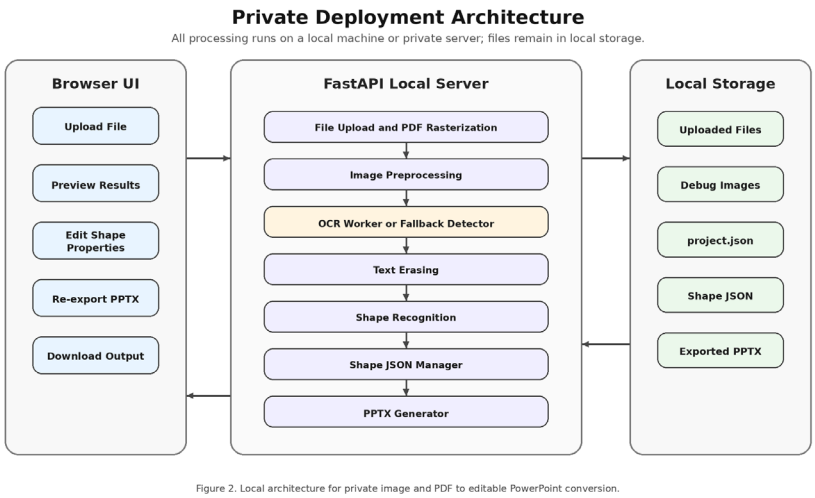


Figure 2. Local architecture for private image and PDF to editable PowerPoint conversion.

3.3. Core Backend Modules

The backend is divided into specialized modules. Each module has a narrow responsibility so that OCR, preprocessing, shape recognition, text removal, and PowerPoint generation can be improved independently.

Table 1. Core backend modules and responsibilities.

Module	Responsibility
File upload module	Receives images or PDF files and stores standardized input files.
PDF rasterization module	Converts PDF pages into images for page-level analysis.
OCR module	Detects text content and text bounding boxes.
Image preprocessing module	Performs denoising, color clustering, and edge detection.
Shape recognition module	Detects rectangles, rounded rectangles, lines, arrows, polygons, color blocks, and dashed boxes.
Text erasing module	Removes detected text regions and fills them with estimated background colors.
Shape JSON module	Stores editable geometry, style, text placeholders, and layer order.
PPTX generation module	Converts Shape JSON into editable PowerPoint shapes.
Debug output module	Exports detection previews, masks, repaired images, and logs.
Task management module	Stores previous processing tasks and generated outputs for private reuse.

4. Methodology

4.1. Image Normalization and Preprocessing

The first processing stage standardizes the input image. For image input, the system reads pixel dimensions and computes the corresponding slide aspect ratio. For PDF input, each page is rasterized into an image before further processing. Large images can be scaled to a processing-friendly size while preserving coordinate mappings to the original resolution.

Preprocessing includes denoising, dominant color estimation, edge detection, and contour extraction. Color clustering is used to identify large uniform areas such as background panels, colored blocks, and layout regions. Edge and contour analysis is used to identify boundaries, rectangular structures, lines, and other geometric elements. These operations prepare the image for text removal and shape classification.

4.2. Text Detection and Text-Region Removal

Text detection is based on OCR when OCR models are available. The OCR module returns text boxes and associated coordinates. For large pages, the system may scale the image before OCR processing and then map the detected text boxes back to the original coordinate system. This reduces processing cost while maintaining alignment with the source page.

The system also includes a fallback detector based on OpenCV heuristics. This fallback is necessary because OCR model initialization can fail, model files may be unavailable, or inference may exceed the acceptable request time. When OCR fails, the system records the failure reason and returns heuristic text-region candidates rather than blocking the entire conversion workflow.

After text boxes are detected, the text erasing module fills each region with an estimated background color. The fill strategy considers the surrounding region, the local shape background, and the page background. This reduces the risk of filling a

colored shape with page white, or filling a white title area with text color. The objective is not semantic text recovery, but clean removal of original text so that the exported PowerPoint file can receive new editable text.

4.3. Shape Extraction and Classification

Shape extraction focuses on major layout elements rather than perfect recovery of every icon, shadow, or decorative detail. The system identifies large color regions through clustering and connected component analysis. Rectangles and rounded rectangles are inferred from contours, bounding boxes, corner patterns, and edge continuity. Horizontal and vertical lines are detected through edge orientation and aspect-ratio constraints. Dashed borders are detected by repeated short contour segments along a common boundary. Simple arrows and polygons are identified through contour approximation and endpoint analysis.

Each detected visual element is converted into a candidate shape record. Candidate records include geometry, color, border attributes, and confidence information. Small noisy regions are filtered unless they are connected to a larger structural element. Overlapping shapes are ordered according to size, position, color continuity, and detected occlusion patterns.

4.4. Shape JSON Intermediate Representation

The Shape JSON representation separates image understanding from PowerPoint generation. This design makes the system easier to debug and edit. The frontend can load Shape JSON, display a shape list, allow the user to modify coordinates or style attributes, and send the modified JSON back to the backend for re-export.

A simplified shape record is shown below.

```
{
  "id": "shape_001",
  "type": "rounded_rectangle",
  "x": 120,
  "y": 85,
  "width": 420,
  "height": 160,
  "fill_color": "#F2C94C",
  "line_color": "#333333",
  "line_width": 2,
  "dash": false,
  "z_index": 3,
  "text_placeholder": true,
  "confidence": 0.86
}
```

This representation can support multiple shape types while preserving a consistent editing model. It also allows later versions of the system to add confidence visualization, manual correction history, template matching, or batch processing without changing the final .pptx generation logic.

4.5. PowerPoint Reconstruction

The final stage converts Shape JSON into a PowerPoint file. The slide size is set according to the input aspect ratio. Pixel coordinates are converted into PowerPoint units. Fill colors, border colors, border widths, dashed-line styles, and layer order are applied to corresponding PowerPoint shapes. Empty text boxes or text placeholders can be inserted so that the user may add new editable text after export. The key requirement is editability. In the generated .pptx, major color blocks and reconstructed shapes should be independently selectable. Users should be able to move, resize, recolor, and add text to these objects directly in PowerPoint.

5. Evaluation Plan and Prototype Results

5.1. Evaluation Scenarios

The system should be evaluated with several practical input types. The first scenario is a simple slide screenshot with large colored blocks and short labels. The second scenario is a PDF page containing a presentation-like layout. The third scenario includes mixed geometry such as rounded rectangles, arrows, lines, and dashed boxes. The fourth scenario includes text over both white and colored backgrounds, which tests whether text-region filling can choose appropriate background colors.

5.2. Evaluation Metrics

Evaluation should measure both visual reconstruction and editability. Visual reconstruction can be measured by comparing detected major shapes with manually labeled major shapes. Let $N_{correct}$ be the number of correctly reconstructed major shapes, and let N_{major} be the total number of manually labeled major shapes. Reconstruction coverage is defined as

$$Coverage = \frac{N_{correct}}{N_{major}} \times 100\% \quad (1)$$

Equation(1) measures whether the system recovers the main structure of the page. Additional metrics include text-removal residual rate, shape editability rate, export success rate, processing time, and manual correction cost. Shape editability rate is especially important because a visually similar slide is not sufficient if the objects cannot be edited independently.

5.3. Evaluation Metrics

The current prototype demonstrates the core conversion loop. It accepts image input and supports PDF pages after rasterization. It generates text-removed previews, detection previews, Shape JSON records, and editable .pptx files. Reconstructed shapes can be selected and modified in PowerPoint. The frontend supports preview switching, shape listing, basic property editing, and re-export.

The current version should be interpreted as an engineering prototype rather than a completed benchmarked system. Its strongest performance is expected on pages with large color blocks, clear geometric shapes, and relatively simple layouts. It is not designed to perfectly restore complex icons, gradients, shadows, photographic materials, or semantic relationships in the first version.

6. Discussion

The proposed system has three main advantages. First, it preserves privacy because processing can run on a local machine or private server. This is important for internal presentations, confidential PDF files, and personal archives. Second, it emphasizes editable reconstruction rather than static visual reproduction. The output is useful because the user can modify shapes and add new text in PowerPoint. Third, the modular design allows gradual improvement. OCR, shape detection, text removal, frontend correction, and PowerPoint generation can be optimized independently.

However, several limitations remain. The reconstruction of complex visual elements is difficult when objects overlap, use shadows, contain gradients, or have low contrast. OCR failure can reduce text-removal quality, although fallback detection reduces complete pipeline failure. Shape classification may also confuse decorative icons with structural geometry. Manual correction is therefore necessary for practical use, especially when automatic detection is imperfect.

The system intentionally excludes several features from the first version. These include multi-user accounts, permission management, cloud sharing, online collaboration, full semantic understanding, perfect icon reconstruction, and a complete web-based PowerPoint editor. Excluding these features keeps the first implementation focused on the core value: converting raster visual structure into editable presentation geometry.

References

[1] Bradski, G. (2000) The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[2] Suzuki, S. and Abe, K. (1985) Topological Structural Analysis of Digitized Binary Images by Border Following. Computer Vision, Graphics, and Image Processing, 30, 32-46.

[3] Smith, R. (2007) An Overview of the Tesseract OCR Engine. Proceedings of the Ninth International Conference on Document Analysis and Recognition, Curitiba, 23-26 September 2007, 629-633. https://doi.org/10.1109/ICDAR.2007.4376991

[4] Du, Y. et al. (2020) PP-OCR: A Practical Ultra Lightweight OCR System. arXiv preprint, arXiv:2009.09941.

[5] Artifex Software, Inc. (2024) PyMuPDF Documentation. https://pymupdf.readthedocs.io/.

[6] python-pptx Contributors (2024) python-pptx Documentation. https://python-pptx.readthedocs.io/.

[7] Ramírez, S. (2024) FastAPI Documentation. https://fastapi.tiangolo.com/.