AI-DRIVEN FILE TRANSFORMATION: ENABLING CONSTRAINT COMPLIANCE FOR IMAGES AND PDFS

Akshata Dharmadhikari
Computer Engineering
DJSCE
Mumbai, India
dharmadhikariakshata@gamil.com

Prerna Jadhav
Computer Engineering
DJSCE
Mumbai, India
jadhavprerna383@gmail.com

Devansh Mehta
Computer Engineering
DJSCE
Mumbai, India
devanshmehta06@gmail.com

Niraj Mistry
Computer Engineering
DJSCE
Mumbai, India
nirajmestry.email@gmail.com

Nancy Nadar
Computer Engineering
DJSCE
Mumbai, India
nancy.nadar@djsce.ac.in

***Abstract*—Managing file size constraints during online form submissions can be tedious, especially for users with limited technical expertise. Existing solutions, such as third-party tools, often introduce privacy risks and cumbersome workflows. This project proposes a Chrome extension that automates file extraction, compression, and reintegration directly within the browser. The extension compresses files such as PDFs, images, and videos while maintaining quality and ensuring compliance with size limits. Techniques like font subsetting and metadata removal optimize compression for diverse file types, including JPEG, PNG, and PDFs.**

Key features include automatic file detection, real-time compression feedback, and user-friendly integration within web forms. The extension also offers additional functionality, such as merging and splitting PDFs and optimizing handwritten notes. Privacy and security are prioritized, with all processing conducted locally or through secure backends to protect sensitive data. To further enhance security, the extension uses AES encryption to safeguard user files during any temporary storage or transfer operations.

Motivated by personal challenges with file size restrictions during job applications, this tool provides a seamless, efficient, and secure solution for users across various platforms. By simplifying a traditionally complex process, the extension enhances the user experience and ensures hassle-free compliance with form requirements.

***Index Terms*—PDF (Portable Document Format), JPEG (Joint Photographic Experts Group), and PNG (Portable Network Graphics), representing the primary file types handled by the Chrome extension. The project leverages advanced algorithms such as K-Means clustering (K-Means Clustering Algorithm) for image compression and DCT (Discrete Cosine Transform) for optimizing file sizes.**

I. Introduction

The digital shift has made online forms indispensable for academic submissions, job applications, and government procedures. However, these platforms impose strict file size limits, compelling users to manually compress or convert files — a complex task for non-technical users. Existing solutions often rely on third-party apps or online tools, raising privacy concerns. Our project proposes a Chrome extension that automates file compression directly within the browser, streamlining the user experience without compromising data security.

We identified that users often face difficulties with file conversions, especially when dealing with high-resolution images, scanned handwritten notes, and multi-page PDFs. The lack of integrated solutions forces users to juggle between various websites and software, increasing the risk of data breaches. Our solution eliminates these pain points by embedding smart compression techniques directly into the browser's workflow, reducing friction and enhancing user convenience.

1) **Automated File Detection:** Automatically detects files attached to online forms, streamlining the upload process.
2) **Real-time Compression:** Provides instant feedback on file size adjustments, helping users make informed decisions.
3) **Enhanced Security:** Local file processing to safeguard user data and privacy.
4) **Handwritten Note Optimization:** Uses machine learning models to enhance the readability of scanned notes while reducing file size.
5) **Cross-Platform Compatibility:** Ensures smooth operation across various web forms and platforms.

II. Literature Review

One notable approach is the Linear Mapping Image Compression (LMIC) algorithm, which leverages the Discrete Cosine Transform (DCT) for adaptive image compression. The method uses 8x8 block division and applies zonal masking adaptively based on the gray-level distance (GLD) of each block. This ensures that high-frequency areas retain essential details, improving overall quality. The algorithm further incorporates genetic optimization to enhance the efficiency of zonal masks, making the compression process adaptable to user-specified ratios while maintaining high Peak Signal-to-Noise Ratios (PSNR) and outperforming traditional JPEG compression in preserving details[1].

Another noteworthy study explores lossless and lossy compression techniques. Lossless methods such as Huffman Encoding and Run Length Encoding (RLE) are widely employed for their ability to preserve data integrity, achieving modest compression ratios between 2:1 and 3:1. On the other hand, lossy techniques like Transform Coding (DCT and DWT) offer higher compression ratios, ranging om 10:1 to 50:1, by discarding perceptually insignificant data. These methods are particularly suitable for multimedia applications, balancing compression efficiency and visual quality.[[2].

The Tetrolet Transform, another innovative wavelet-based approach, adapts to the geometric and textural patterns of images. By using tetromino tiling and selecting the sparsest representation for each 4x4 block, the method minimizes non-zero coefficients, achieving high compression efficiency. This approach is well-suited for applications requiring a balance between compression and high PSNR, particularly in images with intricate patterns.[[3].

JPEG compression, one of the most extensively used methods, applies DCT to transform spatial data into the frequency domain. Its block-based approach splits images into 8x8 segments, enabling selective quantization of high-frequency coefficients to reduce file size. Entropy coding techniques like Huffman or Arithmetic Coding are then employed for efficient encoding. While standard compression ratios range om 10:1 to 20:1 with good visual quality, higher ratios of up to 50:1 can be achieved at the cost of noticeable degradation in quality. This method remains a cornerstone of image compression due to its efficiency and wide adoptions JPEG's compatibility and security; a bitstream-based encryption technique has been introduced [4].

By manipulating DC and AC coefficients and employing image-content-based encryption keys, this approach ensures robust security without significant file size changes. This method highlights the importance of preserving format compatibility while enhancing functionality of the data,[5] hybrid compression methods such as the combination of Burrows-Wheeler Transform (BWT) and Deflate demonstrate superior performance. BWT rearranges data to enhance locality, while Deflate, which combines LZ77 and Huffman Coding, optimizes redundancy removal and encoding. [5].

Together, they achieve compression ratios as high as 57.36, significantly outperforming standalone methods. Such approaches underline the potential of combining multiple algorithms to achieve optimal compression for specific datasets [6], techniques like K-Means clustering and fuzzy logic-based compression illustrate innovative solutions for image compression. K-Means clusters pixels based on their similarity, offering a simple yet effective approach for lossless compression. In contrast, fuzzy logic compression applies fuzzification and defuzzification to adaptively compress image data while preserving visual quality. These methods highlight the versatility of modern image compression techniques in addressing diverse requirements [7], the referenced works collectively emphasize the growing sophistication of compression algorithms. [6].

some adaptive techniques like LMIC and Tetrolet Transform to hybrid methods for text compression, these approaches underscore the critical balance between compression efficiency, quality preservation, and user-specific requirements. By integrating these ideas, file compression can evolve into more accessible, secure, and efficient processes for everyday applications. Image interpolation is a fundamental operation in digital image processing, widely used in resizing, transformation, and compression tasks. Traditional interpolation methods, such as nearest neighbor and bilinear interpolation, are computationally efficient but often fail to preserve image details, leading to artifacts like blurring and aliasing. Advanced methods like bicubic, Catmull-Rom, and Mitchell-Netravali offer improved visual quality but at the cost of higher computational complexity. [7].

Among advanced interpolation techniques, Lanczos interpolation has gained significant attention for its ability to minimize aliasing and preserve high-frequency details. By using a sinc function windowed by another sinc kernel, Lanczos achieves a balance between sharpness and smoothness in the resized images. The number of neighboring pixels considered depends on the kernel order, with higher orders offering superior quality at the expense of increased computational requirements.[8].

Lanczos interpolation is particularly relevant to image compression workflows, where resizing is a critical preprocessing step. Its ability to retain edge sharpness and minimize artifacts makes it ideal for applications requiring high-quality compressed outputs. However, its computational intensity, especially for higher kernel orders, limits its applicability in real-time systems, underscoring the trade-offs between quality and processing speed..

III. Technology Used

Backend	**Frontend**	**Models**	**Database**
Python	Html	NLU	Docker
Flask	CSS	OpenCV	
django	Javascript		

TABLE I
TECHNOLOGIES USED IN THE PROJECT

1) **Backend**:
   - **Python**: It is simple, readable, and has many libraries, making it ideal for backend development.
   - **Flask**: Flask is a very light and flexible web framework for Python, used to develop APIs or web applications. It provides basic tools like routing, request handling, etc., for easy template rendering.
   - **Django**:Django is a high-level Python web framework used for building web applications, while

Docker is a containerization tool that packages apps and their dependencies into containers.

- **Docker**:Using Docker with Django allows for easy deployment, scalability, and consistent environments across development and production.

A. *System Architecture*

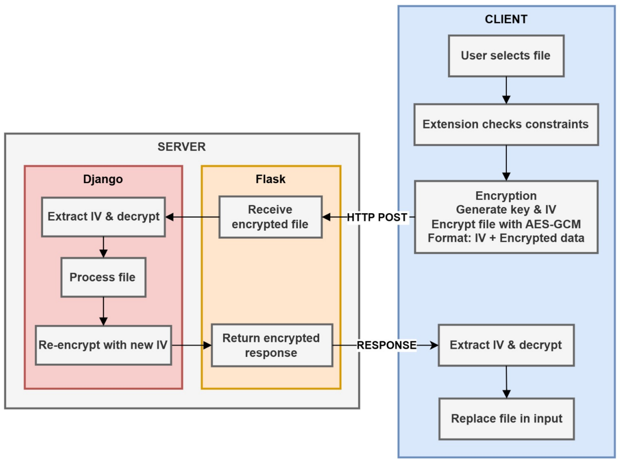


Fig. 1. Secure File Processing Flow

2) **Frontend**:
   - **HTML**: HTML is the standard language used to create and structure content on the web.In this project, HTML is used to build the user interface of the Chrome extension, allowing users to upload files, view compression options, and download processed files.
   - **CSS**:CSS is a stylesheet language used to design and style HTML elements,In this project, CSS ensures a clean, user-friendly interface for the Chrome extension, enhancing user experience by styling buttons, progress bars, and file preview sections while maintaining cross-browser compatibility.
   - **JavaScript**: JavaScript is a versatile scripting language that adds interactivity and dynamic functionality to webpages. In this project, JS handles real-time file size previews, triggers compression actions, and communicates with the backend using APIs.

Models:

1) **OpenCV**:
   - **Haar Cascade Classifier**: This is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images to detect objects (like faces) in images. The pre-trained model is designed to detect frontal human faces.For background removal, the code uses the rembg library, which leverages a deep learning model — typically U-2-Net (a neural network model trained for image segmentation). U-2-Net: A deep learning model that performs salient object detection, isolating the main subject from the background.
2) **Natural Language Understanding (NLU)**:
   - Recognizes user intents (e.g., ”compress this image”) and extracts entities (file names, operations, size limits).
3) **gTTS (Google Text-to-Speech)**:
   - **Emotional Classification**: Converts PDF text into audio files (PDF-to-audio module).
4) **PyPDF2**:
   - **Emotional Classification**: Handles PDF operations — merging, splitting, and querying..

IV. Proposed Methodology

A. *Methodology*

Chrome extension that automates file extraction, compression, and optimization directly within the browser. Advanced techniques like font subsetting, K-Means clustering, and redundant data removal are used to reduce file sizes while preserving quality. The system integrates real-time file size previews and AES-256 encryption to ensure secure and responsive processing. Fig. 1. The diagram illustrates the secure file transmission and processing flow using AES-GCM encryption. User inputs are processed through an intuitive interface, allowing seamless file handling, compression, and format conversion.

B. *System Architecture*

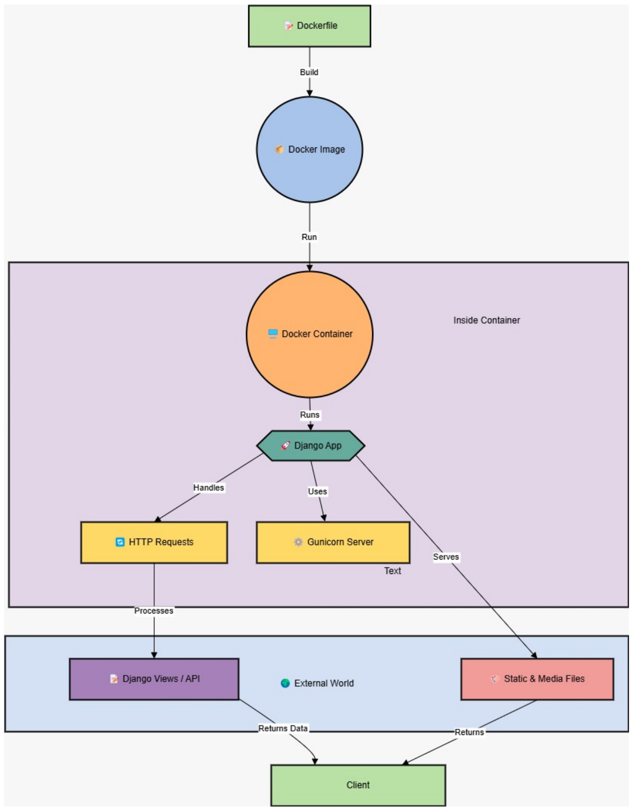


Fig. 2. System Architecture Design.

This diagram illustrates the deployment flow of a Django application using Docker. It starts with a Dockerfile that builds a Docker image, which is then run as a Docker container. Inside the container, the Django app runs, handling HTTP requests and using the Gunicorn server to process and serve responses. The Django app processes views and APIs, interacting with the external world and managing static and media files. Finally, data and responses are returned to the client for display. The design ensures containerized, scalable, and efficient deployment of the Django app.

C. Overview of the System Components

- **1.Flask-based web application:**
  - **File Operations:** Compression, Cropping, Resizing, Conversion (e.g., Excel to PDF), Merging or splitting files, Querying content from documents, Processing notes, These are categorized under the OperationType enum for clarity and validation.
  - **Service Integration:** Relies on external services for certain operations (e.g., image compression, PDF splitting).
  - **AI-Powered Querying:** Uses Google Generative AI (gemini-pro) to analyze instructions and query file contents.
- **Image Compression: :**
  - **Methods Used:** Utilized Discrete Cosine Transform (DCT) for frequency-based compression by dividing images into 8×8 blocks and discarding less important frequency components. Lanczos Interpolation is employed for resizing during optimization, ensuring minimal aliasing and sharp visual quality
  - **Measurements:** Compression effectiveness is evaluated using Peak Signal-to-Noise Ratio (PSNR) to maintain visual fidelity. Users receive real-time file size feedback to guide adjustments based on quality vs. size trade-offs.
- **PDF Compression:**
  - **Methods Employed:** Incorporated Flate-Deflate compression (using LZ77 and Huffman Coding) to compress text and structural elements. Font subsetting is used to embed only required glyphs, while metadata and redundant content are stripped using tools like PyPDF2 and Ghostscript for cleaner, leaner files..
  - **Measurements:** Compression success is measured by comparing file sizes before and after optimization, with average reductions of up to 30 percent, while maintaining layout integrity and readability.
- **Convert PDF files into audio format.**
  - **Methods Employed:** Using ‘PyPDF2‘ to retrieve text from a PDF file. Utilizing ‘gTTS‘ for converting text into Speech through Text-to-Speech (TTS) conversion.
  - **Measurements:** The length of the text affects the duration of the audio. The quality of the audio depends on the quality of the input text.

D. Phase-Wise Explanation

- **Phase 1: File Analysis**
  - The analyzeinstruction() method parses operation instructions to extract required parameters..
  - Parameters are tailored for each operation type (e.g., dimensions for cropping).
- **Phase 2: File Processing**
  - The processfiles() method routes the files and parameters to the correct operation type.
  - External services are called via HTTP requests for certain operations.
  - Example: Calls a URL for merging PDFs, saving the resulting file.
- **Phase 3: Query Processing**
  - Extracts content from uploaded files (PDF, Excel, Word, etc.) using libraries like PyPDF2, openpyxl, and docx.
  - Queries are processed by creating a prompt for the AI model and extracting results from file content..

V. Implementations and Discussions

A. Test Cases

Test Case ID	Input	Expected Output	Actual Output
1	Upload PDF file	Display file with size preview	Display file with size preview
2	Apply lossy compression to JPEG	Reduce file size by 50 percent	File size reduced by 48 percent
3	Convert PNG to JPEG	Generate JPEG file	JPEG file generated
4	Merge two PDF files	Produce combined PDF	Combined PDF produced

TABLE II
File Processing Operations

The table outlines key test cases for file processing operations in the project. It includes uploading a PDF file to check size preview, applying lossy compression to a JPEG with an expected 50 percent size reduction (actual result was 48 percent), converting a PNG to a JPEG to verify format conversion, and merging two PDF files to produce a combined document. Each test case compares expected and actual outputs, ensuring the system functions as intended with minimal discrepancies, confirming the accuracy and reliability of file operations.

Test Case ID	Input	Expected Output	Actual Output
1	Apply font subsetting to PDF	Reduce PDF size by 30 percent	PDF size reduced by 28 percent
2	Remove redundant data from PDF	Smaller optimized PDF	Optimized PDF produced
3	Optimize handwritten note scan	Enhanced and compressed note	Enhanced and compressed note

TABLE III
OPTIMIZATION TECHNIQUESS

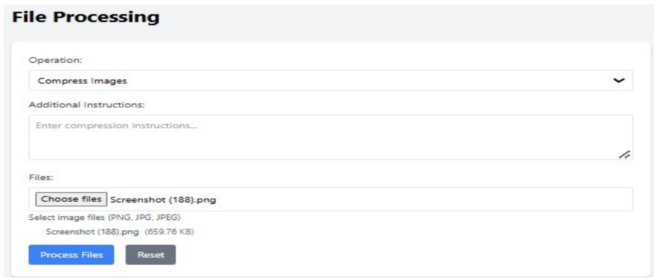
The table presents test cases for advanced file optimization techniques. It includes applying font subsetting to a PDF, aiming for a 30 percent size reduction (achieving 28 percent), removing redundant data to produce a smaller, optimized PDF, and enhancing and compressing handwritten note scans. Each test case verifies that the implemented algorithms — font subsetting, metadata removal, and note optimization — work effectively, ensuring the system reduces file sizes while maintaining content quality.

Test Case ID	Input	Expected Output	Actual Output
1	Real-time file size preview	Display updated file size instantly	Displayed updated file size instantly
2	Encrypt uploaded file	Secure file with AES-256 encryption	File encrypted successfully.

TABLE IV
REALTIME SECURITY TESTS

The table highlights test cases for real-time processing and security features. It tests real-time file size preview, ensuring instant updates when file compression settings are adjusted, and file encryption using AES-256, verifying secure handling of uploaded files. Both test cases confirm the system's responsiveness and data protection capabilities, ensuring efficient feedback and robust security for user files.

B. Screenshots of Outputs



File Processing

Operation: Compress Images

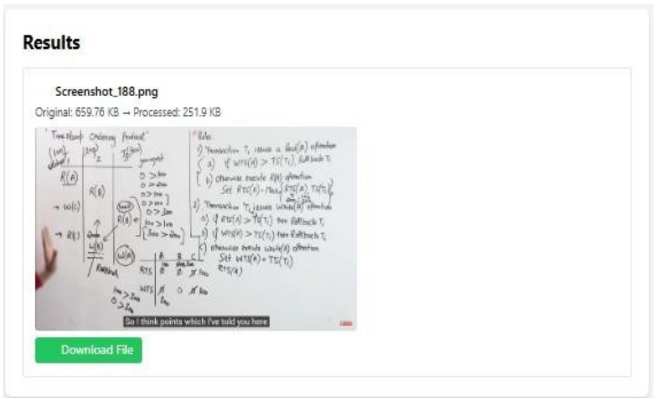
Additional Instructions: Enter compression instructions...

Files: Choose files Screenshot (188).png

Select image files (PNG, JPG, JPEG)

Screenshot (188).png (659.76 KB)

Fig. 3. Compress Image Operation
The image showcases the Chrome extension's interface for image compression. Users can upload an image file, select compression settings (lossy or lossless), and view real-time feedback on file size reduction

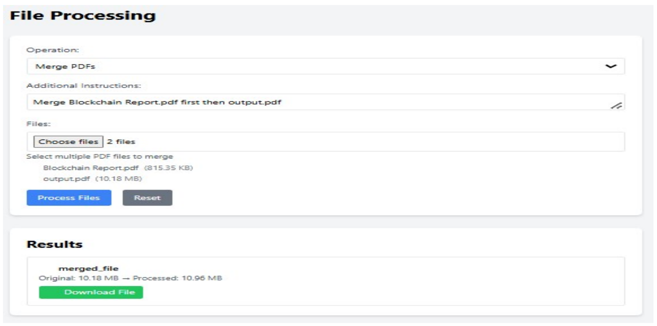


Results

Screenshot_188.png

Original: 659.76 KB → Processed: 251.9 KB

Fig. 4. Compress Image Operation
A progress bar visually indicates the compression process, ensuring transparency and control over the final output.



File Processing

Operation: Merge PDFs

Additional Instructions: Merge Blockchain Report.pdf first then output.pdf

Files: Choose files 2 files

Select multiple PDF files to merge

Blockchain Report.pdf (815.35 KB)

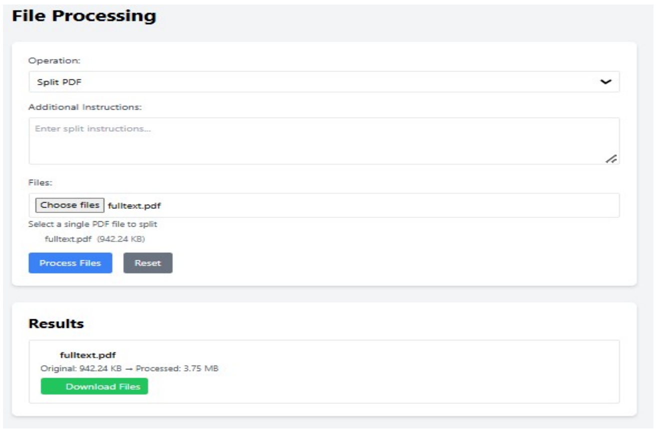
output.pdf (10.18 MB)

Results

merged_file

Original: 10.18 MB → Processed: 10.96 MB

Fig. 5. Merge PDF Operation
This image illustrates the PDF merging feature of the extension. Users can select multiple PDF files for merging, arrange the file order, and preview the combined document.



File Processing

Operation: Split PDF

Additional Instructions: Enter split instructions...

Files: Choose files fulltext.pdf

Select a single PDF file to split

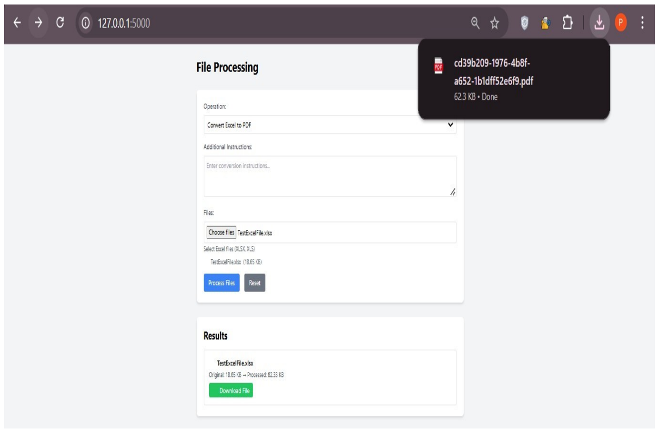
fulltext.pdf (942.24 KB)

Results

fulltext.pdf

Original: 942.24 KB → Processed: 3.75 MB

Fig. 6. Split PDF Operation
The Split PDF interface allows users to upload a multi-page PDF and specify page ranges for splitting. The image displays options for extracting individual pages or separating the document into custom sections.


127.0.0.1:5000

cd39b209-1976-4b8f-a652-1b1dff52e6f9.pdf
62.3 KB • Done

File Processing

Operation: Convert Excel to PDF

Additional Instructions: Enter conversion instructions...

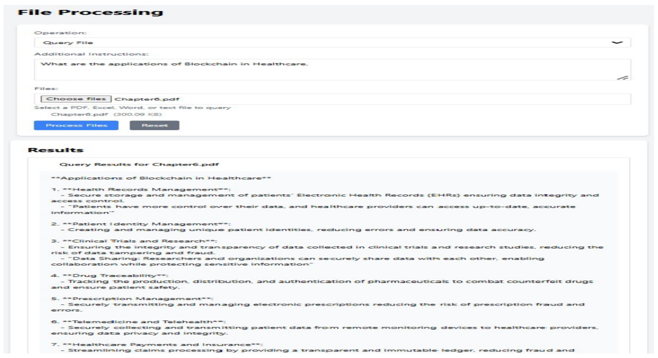
Files: Choose files TestExcelFile.xlsx
Select Excel files (XLSX, XLS)
TestExcelFile.xlsx (18.65 KB)

Process Files | Reset

Results

TestExcelFile.xlsx
Original: 18.65 KB → Processed: 62.33 KB
Download File

Fig. 7. Convert Excel file to PDF file
The screenshot highlights the Excel-to-PDF conversion functionality. Users can upload Excel spreadsheets, preview the tabular data, and customize layout options.


File Processing

Operation: Query File

Additional Instructions: What are the applications of Blockchain in Healthcare.

Files: Choose files Chapter6.pdf
Select a PDF, Excel, Word, or text file to query
Chapter6.pdf (300.09 KB)

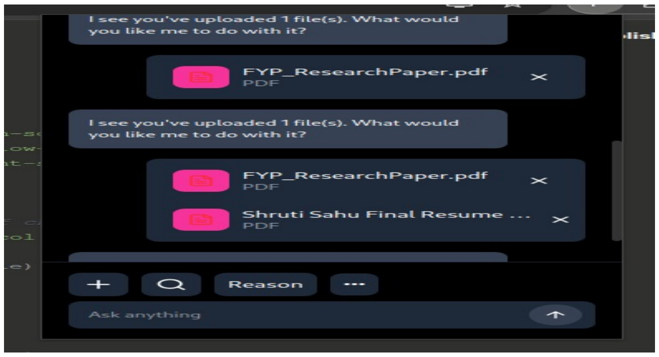
Process Files | Reset

Results

Query Results for Chapter6.pdf

```
**Applications of Blockchain in Healthcare**

1. **Health Records Management**:
  - Secure storage and management of patients' Electronic Health Records (EHRs) ensuring data integrity and
access control.
  - "Patients have more control over their data, and healthcare providers can access up-to-date, accurate
information"

2. **Patient Identity Management**:
  - Creating and managing unique patient identities, reducing errors and ensuring data accuracy.

3. **Clinical Trials and Research**:
  - Ensuring the integrity and transparency of data collected in clinical trials and research studies, reducing the
risk of data tampering and fraud.
  - "Data Sharing: Researchers and organizations can securely share data with each other, enabling
collaboration while protecting sensitive information"

4. **Drug Traceability**:
  - Tracking the production, distribution, and authentication of pharmaceuticals to combat counterfeit drugs
and ensure patient safety.

5. **Prescription Management**:
  - Securely transmitting and managing electronic prescriptions reducing the risk of prescription fraud and
errors.

6. **Telemedicine and Telehealth**:
  - Securely collecting and transmitting patient data from remote monitoring devices to healthcare providers,
ensuring data privacy and integrity.

7. **Healthcare Payments and Insurance**:
  - Streamlining claims processing by providing a transparent and immutable ledger, reducing fraud and
```

Fig. 8. Query the file
This image captures the query feature, where users can input specific questions or keywords to extract information from supported file types (PDFs, Excel, etc.).


I see you've uploaded 1 file(s). What would you like me to do with it?

FYP_ResearchPaper.pdf
PDF

I see you've uploaded 1 file(s). What would you like me to do with it?

FYP_ResearchPaper.pdf
PDF
Shruti Sahu Final Resume ...
PDF

Fig. 9. Query the file
The chatbot allows users to upload files and guides them through simple file management tasks.

VI. Conclusion

The reviewed techniques exemplify significant advancements in data compression, addressing a broad spectrum of real-world challenges with a focus on efficiency, adaptability, and user-centric design. As compression demands grow across domains—from multimedia handling to secure document transmission—these innovations prove critical in ensuring optimal performance without compromising data quality.

The integration of advanced methods such as Linear Mapping Image Compression (LMIC), which combines DCT and genetic algorithms, demonstrates the potential for intelligent, user-defined optimization. Similarly, the Tetrolet Transform, with its wavelet-based adaptive tiling, offers a powerful solution for preserving geometric and textural fidelity in high-precision visual content. These contributions build on traditional foundations, with widely-used standards like JPEG continuing to evolve through security-conscious enhancements such as bitstream-based encryption, ensuring continued relevance in today's digital ecosystems.

Hybrid strategies—like the combination of the Burrows-Wheeler Transform (BWT) with Deflate—underscore the potential of algorithm fusion to achieve exceptional compression ratios, particularly for textual and structured data. Meanwhile, classic lossless methods like Huffman Encoding and Run-Length Encoding (RLE) remain indispensable in contexts where data integrity is non-negotiable. In contrast, lossy approaches such as fuzzy logic-based compression and 2D Discrete Wavelet Transform (2D-DWT) continue to play a pivotal role in multimedia applications, where perceptual quality must be preserved.

These algorithms do not exist in isolation—they power practical, accessible tools like the proposed Chrome extension for automated file compression, which leverages content-aware techniques, real-time feedback, local processing, and robust encryption (e.g., AES-256) to provide a seamless user experience. The inclusion of features like font subsetting, Lanczos interpolation, Flate-Deflate compression, and DCT-based image optimization exemplifies how merging foundational principles with modern advancements results in highly effective, scalable solutions.

In summary, these developments form the backbone of modern compression technology. They not only enhance storage and transmission efficiency but also empower users with tools that are secure, adaptive, and intuitive. As these technologies continue to evolve, they will play an increasingly vital role in overcoming the limitations of digital data management—paving the way for smarter, faster, and more secure workflows across industries.

References

[1] M. R. Bonyadi and M. E. Moghaddam, "A nonuniform high- quality image compression method to preserve user-specified compression ratio," in *International Journal of Image and Graphics, Vol. 11, No. 3, 2011, DOI: 10.1142/S0219467811004123.*, 2011, pp. 314–319.

[2] S. H. P. Jaya S. Kulchandani and K. J. Dangarwala, "Image compression: Review and comparative analysis," in *International Journal of Engineering Research Technology (IJERT), Vol. 3, Issue 11, November 2014, ISSN: 2278-0181.*, 2014, pp. 922–926.

[3] S. A. R. Naqvi, "International conference on open-source systems and technologies," in *Image Compression Using Haar Wavelet-Based Tetrolet Transform*, vol. 41, no. 1, 2013, pp. 173–182.

[4] M. A. E.-D. A. M. Raid, W. M. Khedr and W. Ahmed, “Jpeg compression using discrete cosine transform (dct),” in *International Journal of Computer Science Engineering Survey (IJCSES), Vol. 5, No. 2, April 2014*, 2014, pp. 1–4.

[5] S. T. M. I. J. H. F. I. Junhui He, Shuhao Huang, “Jpeg compatible joint image compression,” in *IEEE (Institute of Electrical and Electronics Engineers) Transactions on Multimedia*, 2020, pp. 1–6.

[6] E. O. Serkan Keskin, Onur Sevli, “Single and binary performance comparison of data compression algorithms,” in *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi (Bitlis Eren University Journal of Science), Vol 3, 2023*, 2023, pp. 314–319.

[7] K. K. Abhipriya Singh, “Review of image compression techniques,” in *Proceedings of the International Conference on Recent Innovations in Signal Processing and Embedded Systems (RISE-2017),*, 2023, pp. 922–926.

[8] D. P. V. V. R. S. Mr. Pankaj S. Parsania1, “A comparative analysis of image interpolation algorithms,” in *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 41, no. 1, 2011, pp. 173–182.