Tribhuvan University

Faculty of Humanities and Social Science

A Project Report on

FILE COMPRESSION SYSTEM

"FilePres"

Submitted to

Department of Computer Application

Academia International College

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted by

Prabhat Gurung

(6-2-346-25-2020)

Neer Bahadur Shrestha

(6-2-346-21-2020)

Nov 12, 2025

Under the Supervision of

Ananda Adhikari

Tribhuvan University

Faculty of Humanities and Social Science

Academia International College

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by "**Prabhat Gurung"** and "**Neer Bahadur Shrestha"** entitled "**File Compression System**" in partial fulfillment of the requirements of the degree of Bachelor of Computer Application is recommended for the final evaluation.

…….

SIGNATURE

Ananda Adhikari

SUPERVISOR

Academia International College

Gwarko, Lalitpur

Tribhuvan University

Faculty of Humanities and Social Science

Academia International College

LETTER OF APPROVAL

This is to certify that this project is prepared by " **Prabhat Gurung** " and " **Neer Bahadur Shrestha**" entitled "**File Compression System**" in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. **Mr. Ananda Adhikari** **Supervisor** **Academia International College** **Gwarko, Lalitpur**	**……..** **Mr. Bishwas Mathema** **Coordinator** **Academia International College** **Gwarko, Lalitpur**
…….. **Internal Examiner**	**……..** **External Examiner**

STUDENT'S DECLARATION

We hereby declare that we are the only author of this work and that no sources other than those listed have been used in this work.

…….	…….
Prabhat Gurung	Neer Bahadur Shrestha
Date:	Date:

ABSTRACT

This project presents a web-based file compression system focused on improving file storage and transfer efficiency. It supports client-side image compression and server-side PDF compression, providing real-time quality comparison, file size visualization, and detailed compression statistics. The platform includes a dual-access model: guest users can perform basic compression, while authenticated users gain access to advanced features such as compression history, file management, and file sharing. The system offers an intuitive interface with adjustable quality controls and before-after previews. This report outlines the full development process, covering requirement analysis, system architecture, implementation, and testing, demonstrating the practical application of software engineering principles to address real-world file optimization needs.

Keywords: File Compression, Image Compression, PDF Compression, Web-Based System, File Optimization.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Academia International College for giving us the opportunity to work on this project titled **"File Compression System."** We are especially grateful to our supervisor, **Mr. Ananda Adhikari**, for his valuable guidance, support, and encouragement throughout the project. His insights and feedback greatly enhanced our understanding and enabled us to complete the work successfully.

We also wish to thank our teachers and friends for their continuous support, motivation, and cooperation. Their assistance boosted our confidence in overcoming challenges and completing the project on time. We are sincerely thankful to everyone who contributed, directly or indirectly, to the successful completion of this project.

Prabhat Gurung

TU Exam Roll No: 6-2-346-25-2020

Neer Bahadur Shrestha

TU Exam Roll No: 6-2-346-21-2020

Table of Content

LIST OF ABBREVIATIONS

API	Application Programming Interface
CSS	Cascading Style Sheet
HTML	Hyper Text Markup Language
JPEG	Joint Photographic Expert Group
JWT	JSON Web Token
ORM	Object-Relational Mapping
PDF	Portable Document Format
PNG	Portable Network Group
REST	Representational State Transfer
SQL	Structured Query Language
TS	TypeScript

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

1.1 Introduction

The File Compression System is a web-based platform designed to make image and PDF compression simple, efficient, and accessible. It helps users reduce file sizes for faster sharing and improved storage management without compromising essential quality. The platform offers two usage levels: guest users can perform basic compression tasks, while registered users gain access to advanced features such as compression history, file management, and file sharing, allowing them to track and organize their compressed files over time.

The system performs image compression directly in the browser, ensuring fast processing, reduced server load, and immediate user feedback. Users can adjust quality settings, preview results instantly, and compare original and compressed versions side-by-side to make informed decisions about quality and size. By providing a responsive interface and streamlined workflow, this project delivers an installation-free solution suitable for both casual users and professionals who need consistent, reliable, and user-friendly file compression capabilities.

1.2 Problem Statement

With the increasing use of digital media in personal and professional contexts, users frequently encounter challenges related to file storage and transmission. High-resolution images and large PDF documents consume significant storage space and create barriers in scenarios with limited bandwidth or storage capacity. Email attachments often have size restrictions, cloud storage services impose usage limits, and mobile data constraints make large file transfers expensive and time-consuming.

Existing compression solutions often require software installation, lack intuitive interfaces, or do not provide adequate control over the compression process. Many tools operate as black boxes, leaving users uncertain about the quality-size tradeoff they are making. Additionally, users who regularly compress files lack a centralized system to track their compression history, manage compressed files, and share them efficiently.

The primary problems this system addresses include:

- Limited accessibility to compression tools without software installation requirements.
- Absence of visual comparison tools to evaluate compression results.
- No centralized platform for managing compression history for frequent users.

1.3 Objectives

- To implement efficient image and pdf compression that reduces file sizes.
- To implement a secure authentication system that differentiates between guest and registered users.
- To develop a compression history tracking system for authenticated users.

1.4 Scope and Limitation

1.4.1. Scopes:

Scopes are as follows:

- User registration and authentication system with secure password handling.
- Guest access with limited compression capabilities.
- Client-side compression using for Images with adjustable quality slider for user-controlled compression levels and interactive comparison slider.
- Pdf compression with quality selection options.
- Batch processing capabilities for multiple files.
- History of all compression operations for authenticated users with download functionality.

1.4.2. Limitations:

Limitations are as follows:

- The system currently only supports image formats(JPEG, PNG, JPG, WebP) and Pdf files and other document formats are not supported yet.
- Client-side image compression requires modern browser support and adequate client-side processing power.
- Both original and compressed files are stored on the server and not a dedicated file storage.

1.5 Development Methodology

The FilePres project was developed using the agile methodology, specifically Scrum, which provided the flexibility needed to accommodate evolving requirements throughout the development lifecycle. Agile allowed us to respond quickly to feedback, prioritize features effectively, and deliver working increments of the application at regular intervals. The development process was organized into two-week sprints, each beginning with sprint planning and ending with a sprint review and retrospective. Key Agile/Scrum practices that was followed daily during the development process were daily stand-up meetings discussing on what we accomplished and working on currently, issues encountered, helping in identifying issues early and solve the issues together. Features were built, tested and improved upon feedbacks ensuring that all components and services worked properly before moving forward. By adhering to these agile practices, we were able to deliver a high-quality, user-created application while remaining adaptable to new requirements, bug reports, and feedbacks throughout the entire development cycle. This iterative and collaborative approach significantly contributed to the successful and on-time delivery of FilePres.

1.6 Report Organization

1) **Introduction:**

This chapter provides an overview of the project, establishing the context for file compression in modern digital workflows. It presents the problem statement that motivated the development, outlines the specific objectives the system aims to achieve, and defines the scope and limitations of the current implementation.

2) **Background Study and Literature Review:**

This chapter explores the theoretical foundation of file compression technology, examining various compression algorithms and their applications. It reviews existing compression solutions and academic research in the field, identifying gaps that this project addresses.

3) **System Analysis and Design:**

This chapter presents the detailed analysis conducted during the project planning phase. It documents requirements, feasibility analysis, and data modeling through ER diagrams.

It also covers process modeling using Data Flow Diagrams (DFD) and presents the complete system design.

4) Implementation and Testing:

This chapter describes the practical realization of the system design. It details the tools and technologies employed, provides implementation specifics for each module, and presents comprehensive testing strategies including unit testing and system testing with detailed test cases and results

5) Conclusion and Future Recommendation:

This chapter summarizes the achievements of the project, evaluates the extent to which objectives were met, and discusses lessons learned during development. It also proposes future enhancements including support for additional file formats, advanced compression algorithms, and feature expansions based on user feedback.

Chapter 2: Background Study and Literature Review

2.1 Background Study

File compression technology has been widely researched due to its importance in reducing storage requirements and improving data transfer efficiency. Earlier file management systems stored data in its original form, which consumed large amounts of space and caused bandwidth limitations, especially for users with restricted storage or slow internet. Research from platforms such as Google Scholar, IEEE Xplore, and ResearchGate consistently shows that uncompressed files create significant inefficiencies as digital content continues to increase.

Studies highlight that modern compression algorithms fundamentally changed how digital files are stored and shared. Lossless methods like DEFLATE, LZ77, and Huffman coding preserve full data integrity and are essential for documents and software files, while lossy techniques such as JPEG and MP3 achieve higher compression ratios by removing less critical data, making them ideal for multimedia. Research on web-based systems also emphasizes the advantages of client-side processing enabled by HTML5 APIs, allowing tools like browser-image-compression to perform fast in-browser compression with instant previews, reduced server load, and no need to upload large originals.

Academic literature further explains optimization techniques for PDFs such as stream compression, image down sampling, and font subsetting and shows how compression effectiveness varies with document content. Meanwhile, secure authentication practices, including JSON Web Tokens and hashing algorithms like bcrypt or Argon2, are widely recommended for modern web applications. Combining these algorithms, client-side and server-side processing, and strong security principles results in efficient, scalable, and user-friendly compression systems capable of meeting real-world data management needs.

2.2 Literature Review

Image compression has emerged as a critical area in data management, particularly with the exponential growth of digital visual content in fields ranging from general computing to bioinformatics. Early efforts focused on foundational techniques to reduce storage and transmission overhead while preserving essential information. One pioneering exploration in this domain is the chapter on digital image compression within the Handbook of

Weighted Automata, which lays groundwork for algorithmic approaches to compressing visual data by integrating automata theory with image processing principles. Although detailed methodological specifics are limited in accessible previews, the work highlights the potential of weighted models for efficient encoding, setting a theoretical stage for subsequent advancements in lossless and lossy paradigms. [1]

Building on theoretical foundations, comprehensive surveys have synthesized decades of progress in lossless image compression, emphasizing entropy-based and dictionary methods to eliminate redundancy without data loss. For instance, Rahman and Hamada (2019) provide an in-depth state-of-the-art review, analyzing algorithms such as run-length encoding (RLE), Shannon-Fano, Huffman, Lempel-Ziv-Welch (LZW), and arithmetic coding through unified numeric examples and benchmarks on medical images. Their findings reveal arithmetic coding's superior compression ratios (up to 71.25% storage savings) but highlight Huffman's practical balance in speed and efficiency, underscoring the role of probabilistic models and binary tree structures in exploiting data symmetry for optimal encoding. This survey not only benchmarks performance metrics like average code length and peak signal-to-noise ratio but also exposes vulnerabilities, such as error sensitivity in Huffman, informing hybrid designs for real-world applications. [2]

Complementing these technical deep dives, broader literature reviews have cataloged the evolution of both lossless and lossy techniques, addressing their applicability across diverse computational contexts. Al-jawaherry and Hamid (2021) offer a systematic overview, classifying methods into predictive coding, wavelet transforms, discrete cosine transforms (DCT), fractal-based approaches, and entropy encoding, drawing from hybrid innovations like DCT with vector quantization. Their analysis emphasizes lossless variants (e.g., JPEG-LS) for error-free reconstruction in sensitive domains and lossy ones (e.g., singular value decomposition optimized by particle swarm) for high-ratio scenarios, while noting emerging trends in neural networks and compressed sensing for specialized imagery such as medical or astronomical data. This work underscores the ongoing need for adaptive algorithms that balance quality preservation with bandwidth efficiency, synthesizing insights from prior studies without introducing novel implementations. [3]

While general image compression dominates early literature, parallel developments in digital libraries have explored compression enhancements for document formats, extending principles to non-visual data structures. A contribution in this vein, documented in the

European Digital Mathematics Library, examines PDF enhancements and tools tailored for archival systems, focusing on lossless strategies to maintain document integrity amid growing repository sizes. Though specifics on combinatorial optimizations are sparse in available excerpts, the paper advocates for integrated compression workflows that leverage structural redundancies in PDF streams, influencing broader discussions on scalable data handling in academic and library informatics. This perspective bridges traditional image techniques with document-centric compression, highlighting shared challenges in redundancy reduction. [4]

Advancing into bioinformatics, where high-throughput sequencing generates vast image-like genomic datasets, specialized compression tools have adapted general principles to biological formats. Bonfield (2014) introduces a FASTQ compression utility in Bioinformatics, optimizing for sequence reads, identifiers, and quality scores through context-aware encoding that exploits genomic redundancies. The method employs finite-state models and arithmetic progression to achieve significant size reductions (often 4:1 ratios) while supporting rapid access for downstream analyses, outperforming standard tools like gzip in domain-specific benchmarks. Key innovations include stream decoupling for targeted compression, demonstrating how graph-inspired data structures—such as de Bruijn graphs implicit in sequence modeling—enhance efficiency in handling variable-length biological data, thus extending image compression heuristics to terabase-scale challenges. [5]

Recent hybrid innovations further refine these adaptations, particularly for next-generation sequencing outputs, by combining domain-specific and general-purpose compressors. In their 2019 Source Code for Biology and Medicine article, the authors present MZPAQ, a tool that decouples FASTQ files into identifier, sequence, and quality streams, applying MFCompress (with finite-context arithmetic coding) to the former two and ZPAQ (context-mixing) to quality scores for superior ratios up to 17:1. Evaluated on diverse benchmarks including Illumina and PacBio data, MZPAQ demonstrates robustness across platforms, surpassing competitors like LFQC and Leon in compression depth while regenerating omitted comment lines during decompression. This work reveals trade-offs in speed and memory but affirms the value of algorithmic hybridization—rooted in entropy and dictionary symmetries—for sustainable storage of raw genomic data, paving the way for scalable bioinformatics pipelines. [6]

Chapter 3: System Analysis and Design

3.1 System Analysis

The analysis phase for the file compression system encompassed requirement gathering, feasibility studies, and modeling of data and processes. The analysis identified key stakeholders including end users who need file compression capabilities, system administrators who manage the platform, and developers who maintain and extend the system. Through stakeholder interviews and market research, we established that users primarily need a simple, fast, and reliable compression solution that provides control over quality settings and maintains a history of compressions for future reference.

The existing problem of scattered compression tools, lack of integration between compression and file management, and absence of historical tracking informed the requirements specification. Users expressed frustration with tools that require multiple steps across different platforms to compress, store, and share files. This insight drove the decision to integrate compression, history, and sharing features within a single unified platform.

Figure 3.1: Agile Methodology

3.1.1. Requirement and Analysis

Requirement analysis is of two types:

- Functional Requirements.

- Non-Functional Requirements.

i. Functional Requirement

This section provides the requirement overview of the system. Functional requirement of the system has been listed below:

- The system shall allow users to register with email and password or google account.
- The system should support guest access without registration.
- The system should accept image and pdf uploads and compress.
- The system shall save compression history for authenticated users.

ii. Non-Functional Requirement

Non-functional requirement of the system has been listed below:

- The system should perform compression operations without delays.
- The system should protect Compression data from unauthorized access.
- The system should have a simple, user-friendly interface.
- The system should manage Database schema changes through migration commands.

3.1.2. Feasibility Analysis

i. Technical Feasibility

The technical feasibility of this project depends on whether the required tools are available and capable of supporting the system. The chosen technologies Next.js, TailwindCSS, Express, PostgreSQL, and Prisma are open-source, stable, and widely used. Next.js enables fast, scalable interfaces, Express manages backend logic effectively, PostgreSQL ensures secure data storage, and Prisma provides efficient, type-safe database access. With strong documentation and broad platform support, the project is technically feasible.

ii. Operational Feasibility

Operational feasibility evaluates whether the system is easy to use and maintain. Its web-based design requires no installation, offers an intuitive interface, and supports features like compression history. Clear navigation eliminates the need for training, and the modular code ensures straightforward maintenance and updates. The system is operationally feasible and user-friendly.

iii. Economic Feasibility

Economic feasibility assesses whether the benefits outweigh the costs. As an academic project, expenses are minimal, with developer time as the main investment. The use of free, open-source technologies and tools keeps costs low. Given these factors and the value provided, the project is economically feasible.

iv. Schedule Feasibility

Schedule feasibility assesses whether the project can be completed on time. Adequate time was allocated for analysis, design, implementation, testing, and documentation, with clear task dependencies and sequencing. Proper planning and teamwork ensured the project was completed within the deadline, making it schedule feasible.

- **Gantt Chart**

Gantt chart is a graphical depiction of a project schedule. It is widely used in project management as it helps to display activities at different phases of time. The Gantt chart in our project was created using team Gantt as shown in the figure below.

	File Compression System Gantt Chart									
Task	**Week 1**	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**	**Week 10**
Project Planning										
Requirement Gathering										
Setup Express and Database Schema										
Setup NextJs and TailwindCSS										
Build Project Layout UI Components										
Setup user authentication and JWT										
Image compression implementation										
Pdf compression implementation										
Testing and Debugging										
Documentation and Final Review										

Figure 3.2: Gantt chart of File Compression System

3.1.3. Data Modelling (ER-Diagram)

In the Entity-Relationship Diagram above, there are four entities named user, compression, file, and share. The attributes that are primary keys are underlined. User has attributes like userId (primary key), name, email, passwordHash, provider, providerId, createdAt, and updatedAt. This entity stores information about registered users in the file compression system. Compression has attributes like compressionId (primary key), userId, ratio, originalFileId, compressedFileId, createdAt, and updatedAt. This entity records details about compression operations performed by users, including the compression ratio and references to both the original and compressed file versions. File has attributes like fileId (primary key), name, fileSize, fileType, filePath, createdAt, and

updatedAt. This entity stores metadata about files that are uploaded, processed, or generated through compression operations. Share has attributes like shareId (primary key), fileId, token, count, expireAt, isActive, createdAt, and updatedAt. This entity manages file sharing functionality, tracking shared links, their expiration dates, access counts, and active status.

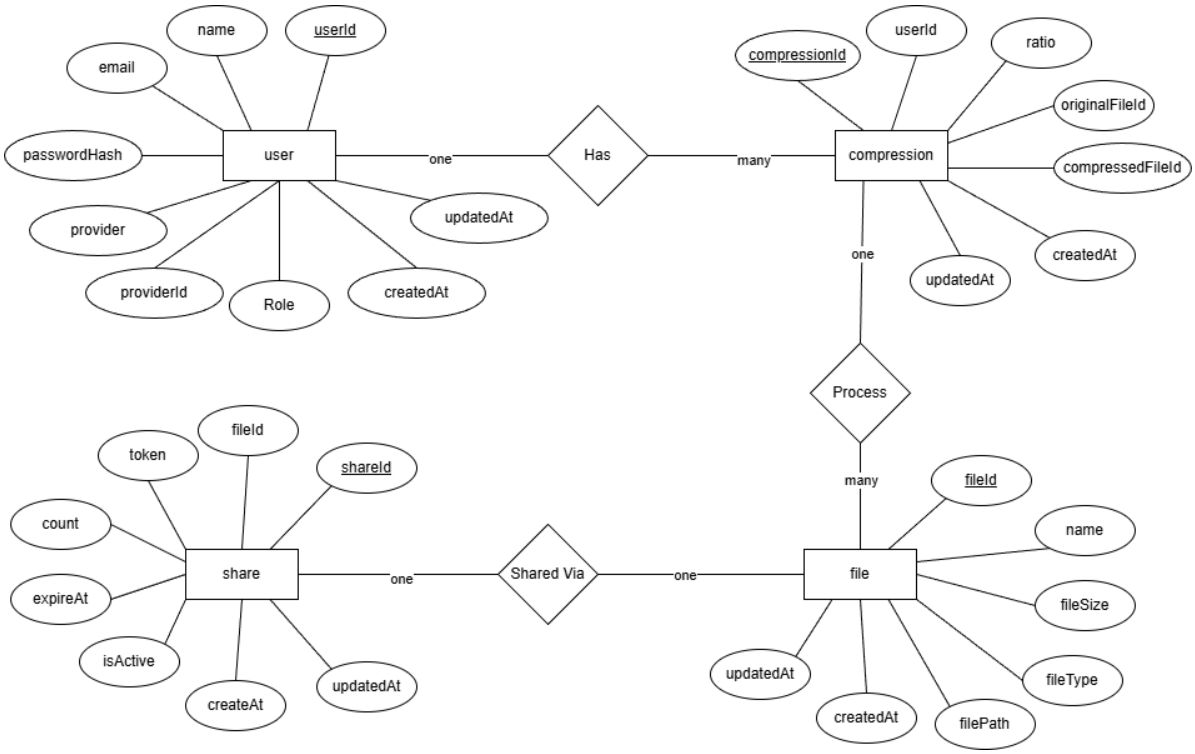

Figure 3.3: ER Diagram of File Compression System

3.1.4. Process Modelling (DFD)

A Data Flow Diagram (DFD) is used to represent the flow of data through the file compression system. Level 0 and Level 1 of the DFD diagram is presented below. The topmost level, known as the context diagram, offers a high-level view of the system and its interactions with external entities.

i. Context Level DFD (Level 0)

In a level 0 DFD, the user is depicted as an external entity, representing the system's interaction with its users. The user interact with the FilePres (File Compression System) process. It handles the compression functionalities.

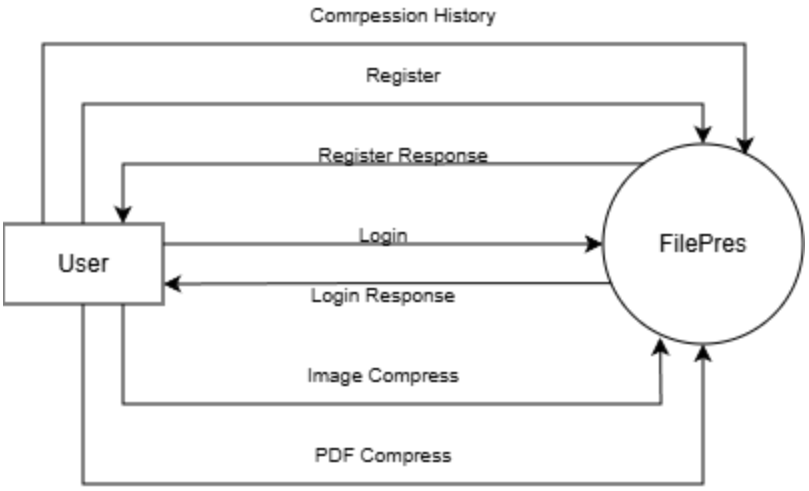

Figure 3.4: DFD Level-0 of File Compression System

ii. Level-1 DFD

In the level-1 DFD below, the user remains as the external entity representing the user of the system. There are 4 new process i.e. User Management, Image Compressor, Pdf Compressor and Compression History. The User Management process handles the user authentication. The Image and Pdf Compressor handles the compression functionality and Compression History handles compression history of the authenticated user.

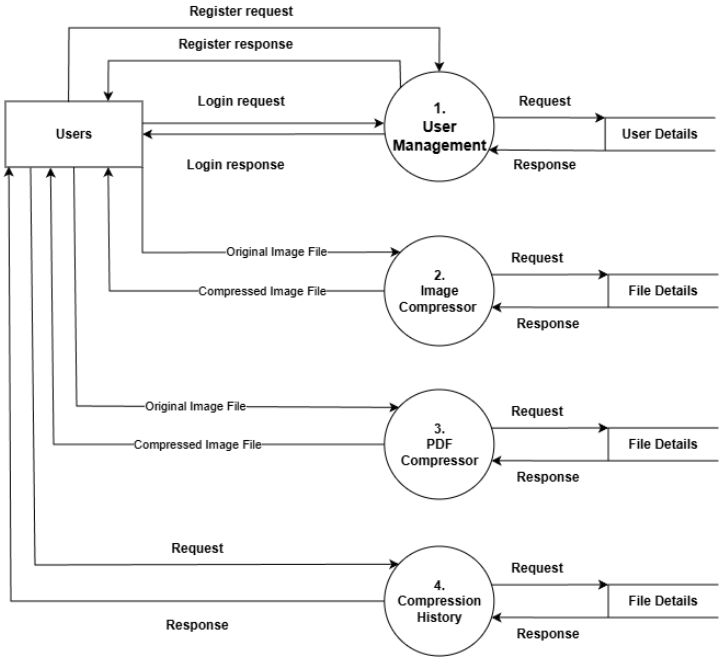

Figure 3.5: DFD Level 1 of File Compression System

3.2 System Design

3.2.1. Architectural Design

FilePres architectural design can be described using a three-tier architectural model which is also known as the client-server architecture. This model separates the application into three main components: presentation layer, application layer, and the data layer. Here is how each layer is organized:

- Presentation Layer (Client-Tier):
  User interact with the presentation layer to initiate image or pdf compression. In NextJs, the pages and components directly handles the presentation layer. Components responsible for displaying UI and gathering user input resides in this layer.
- Application Layer (Middle-Tier):
  The API route are created in the express backend as endpoints that process incoming requests from the frontend. The authentication process like integrating JSON web tokens and OAuth-Based authentication can be seen as part of this layer. This is where services, validation, and business logic are managed such as pdf compression.
- Data Storage Layer (Data-Tier):
  Data storage layer is responsible for storing, retrieving, managing data in a structured and organized manner. In express, we interact with database using the ORM like Prisma.

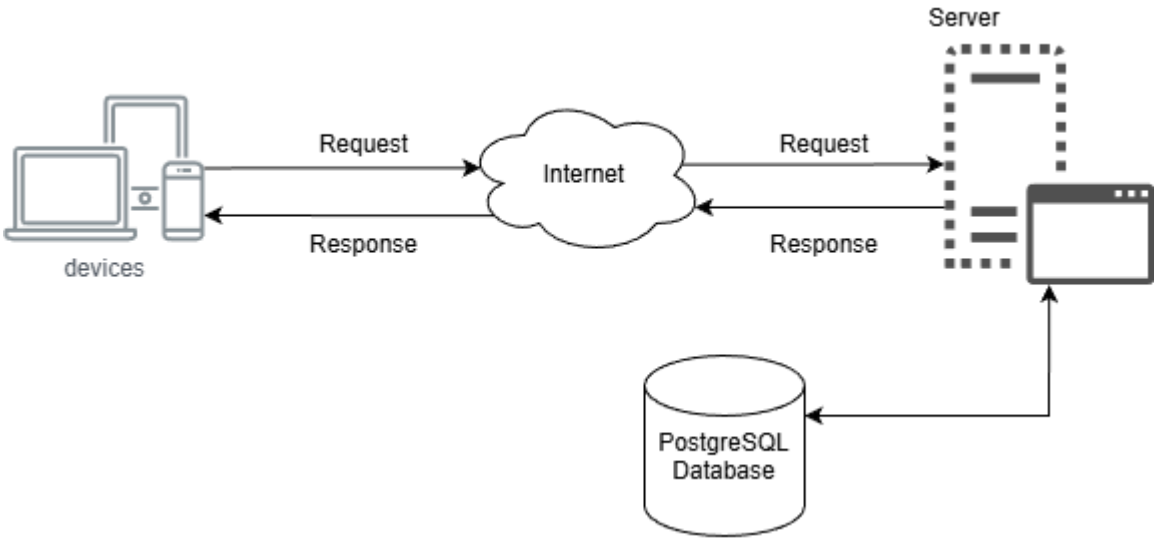

Figure 3.6: Architectural Design of File Compression System

3.2.2. Database Schema Design

In the File Compression System, the database is designed using Prisma ORM with a PostgreSQL database, storing user credentials, file metadata, and compression history.

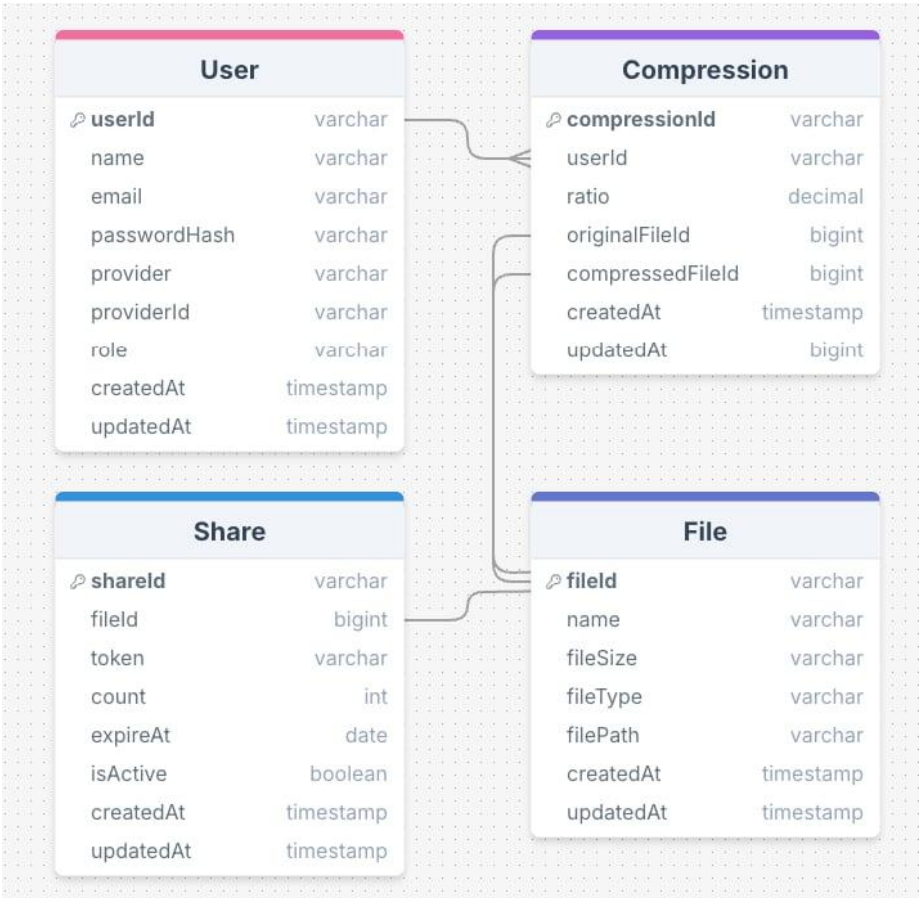

Figure 3.7: Database Schema Diagram of File Compression System

3.2.3. Interface Design (UI Interface / Interface Structure Diagram)

Interface Design also known as UI design, is used to design how File Compression System will look like. The UI design of the login page, image compression page, and pdf compression page are shown below:

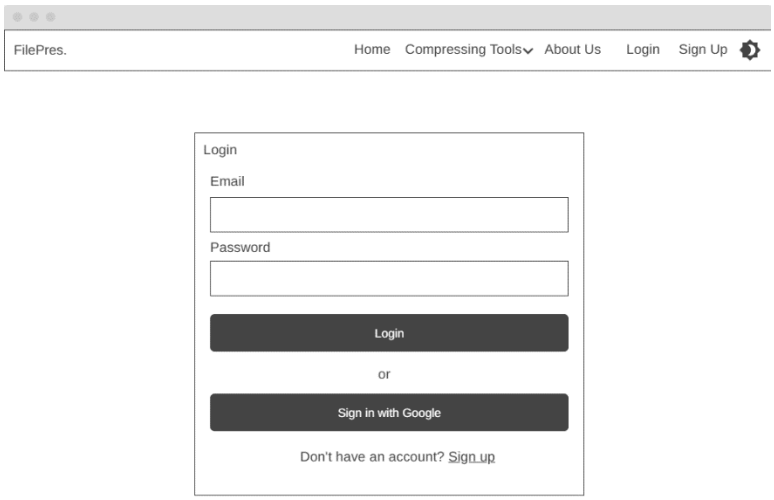

Figure 3.8: Login Page of File Compression System

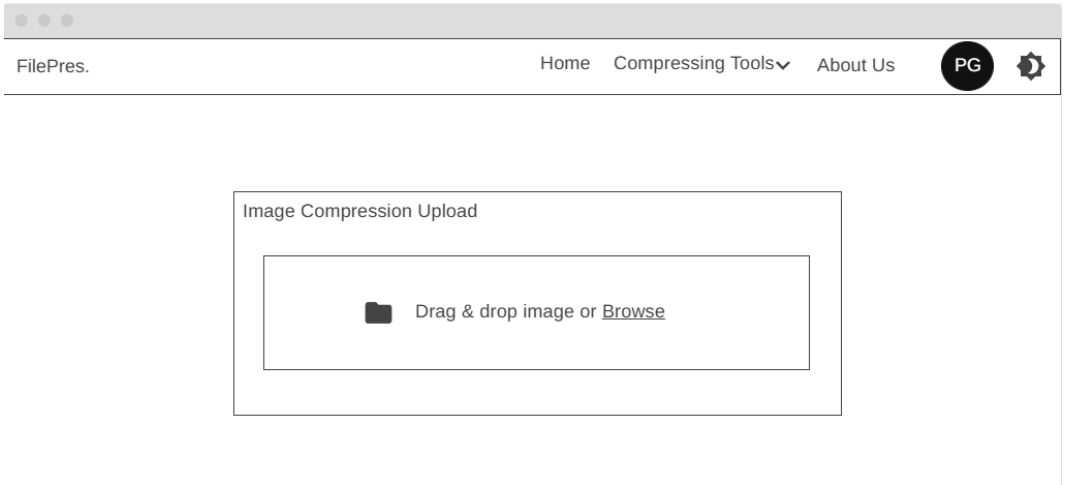

Figure 3.9: Image Compression Interface of File Compression System

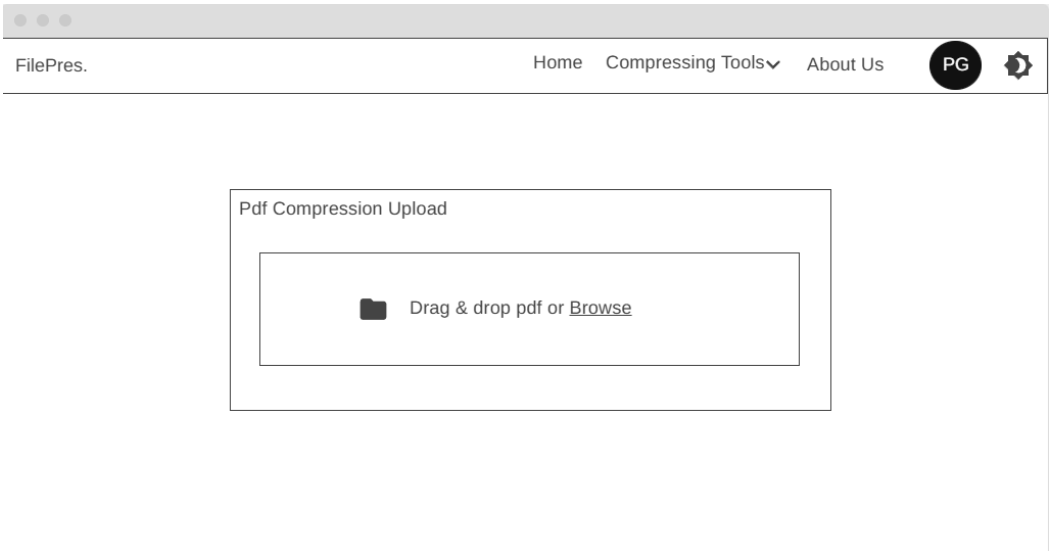

Figure 3.10: Pdf Compression Interface of File Compression System

3.2.4. Physical DFD

A physical DFD is used for graphical representation to depict the physical flow of data within File Compression System. Unlike logical DFD, which focuses on logical flow of data, physical DFD provides a more detailed view of how data moves within the system, considering the physical components involved.

a) User:

The User Physical DFD represents how the data flows between different components within RentEase from the perspective of a regular user. The figure below represents the Physical DFD for Users.

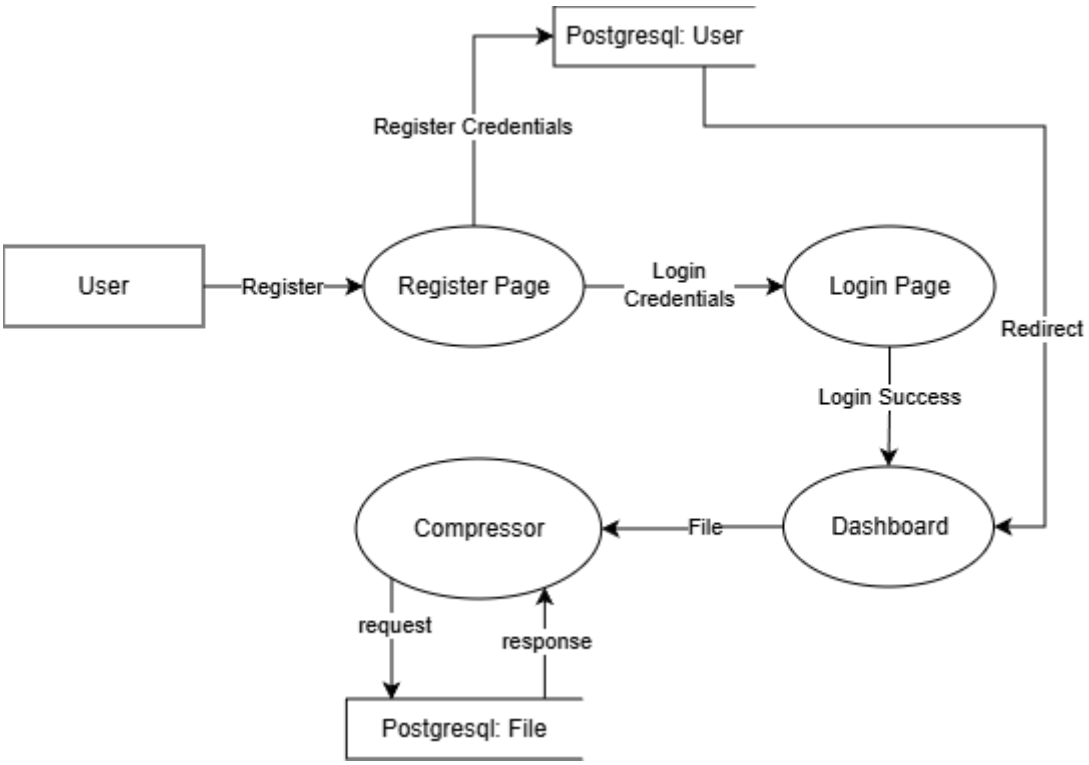

Figure 3.11: User Physical DFD for File Compression System

3.3 Algorithm Details

Description of Algorithm:

In the File Compression System, algorithms are used to compress images and pdf files.

a) DCT (Discrete Cosine Transform)

DCT is the mathematical transformation that converts spatial image information into frequency domain data, allowing for efficient compression by concentrating visually significant information in fewer coefficients:

Steps:

- Color Space Conversion: Convert RGB to YCbCr format.
- Block Division: Divide the image into 8x8 or 16x16 blocks.
- DCT Transform: Apply 2D DCT to each block.
- Quantization: Divide the DCT matrix by a quantization matrix and round the values to remove high-frequency components.
- Entropy Encoding: Use Huffman or arithmetic coding to compress the quantized coefficients.

$$C(u,v) = \sqrt{\frac{1}{N}} for\ u = 0, 0 \leq v \leq N-1$$

$$c(u,v) = \sqrt{(\frac{2}{N})}\ cos\left[\frac{(2v+1)u\pi}{2N}\right] for\ 1 \leq u \leq N-1, 0 \leq v \leq N-1$$

b) Bicubic Interpolation:

Bicubic interpolation is an extension of cubic spline interpolation for interpolating data points on a two-dimensional regular grid, producing smoother surfaces than bilinear or nearest-neighbor methods.

Steps:

- Grid Selection: Bicubic interpolation considers 16 pixels in a 4x4 arrangement around the target position.
- Weight Calculation: For each of the 16 pixels in the neighborhood, calculate a weight based on the distance from the center pixel.
- Apply Cubic Convolution Kernel: Calculate interpolated value using the kernel function.
- Value Estimation: The value of the new pixel is calculated by summing the product of each pixel's value and its corresponding weight, then dividing by the sum of all weights.

$$W(x) = (a+2)|x|^3 - (a+3)|x|^2 + 1\ for\ |x| \leq 1$$

$$W(x) = a|x|^3 - 5a|x|^2 + 8a|x| - 4a\ for\ 1 < |x| < 2, and\ 0$$

Chapter 4: Implementation and Testing

4.1 Implementation

The File Compression System was developed as a web-based application using open-source tools and technologies. Its implementation combines frontend, backend, and database components to ensure efficient, secure, and reliable operation.

4.1.1. Tools Used (CASE tools, Programming languages, Database platforms)

Tools used for the development of this project are as follows:

Frontend

- **NextJs**

  Next.js is a popular React framework used for building server-rendered applications with enhanced performance and flexibility. Next.js simplifies the development process by offering built-in routing, optimized image handling, and API routes, making it a powerful choice for developing full-stack React applications. Its hybrid nature allows developers to choose between static or dynamic rendering, depending on the project's needs.

- **Tailwind CSS**

  Tailwind CSS is a utility-first CSS framework that allows developers to create custom designs quickly and responsively by composing classes directly in their markup.

- **TypeScript**

  TypeScript is a strongly typed superset of JavaScript that enhances code reliability and maintainability by introducing static type checking. It helps catch errors during development and improves overall code quality, making the application more scalable and easier to manage.

Back-End

- **Express**

  Express is a lightweight and flexible Node.js framework used for building backend APIs and server-side applications. It provides a simple yet powerful set of features for routing, middleware handling, and request processing, making it ideal for creating efficient RESTful APIs for web applications.

- **Prisma**

  Prisma is a modern TypeScript ORM that simplifies database access by providing an intuitive and type-safe query builder. It eliminated the complexity of writing raw SQL queries and ensures better consistency, faster development, and automatic type support across the entire backend.

- **JWT**

  JSON Web Tokens (JWT) enables secure user authentication by generating encrypted tokens that verify user identity. These tokens are used to protect API routes, manage sessions, and ensure only authorized users can access restricted areas of the system.

- **Multer**

  Multer is a middleware for handling file uploads in Express applications. It supports uploading single or multiple files efficiently and securely, making it essential for features like uploading documents, images, or compressed files in the system.

Database

- **PostgreSQL**

  Postgresql is a powerful open-source relational database known for its reliability, data integrity, and advanced feature set. It supports complex queries, transactions, and scalability, making it a robust choice for managing structured data in modern web applications.

Development Tools

- **VS Code**

  VS Code was used as the code editor for writing and managing the source code.

Documentation

- **Draw.io**

  Draw.io was used to create diagram like ER and DFD for the system analysis and design of File Compression System.

- **Drawsql.app**

  It was used to display graphical representation of all the tables used in the database.

- **Excel**

  Gantt chart was created on Excel for project planning.

4.1.2. Implementation Details of Modules (Description of procedures/functions)

User Module:

In the user module, an account is created by filling the register form details which include name, email and password or alternatively use google provider to sign up using google id.

- signUp(name, email, password) – passes name, email, password to the backend
- signup(req, res) – get the params and hash the password using Bcrypt, create user on db and generate jwt token and sets the token on res.
- loginWithEmailPassword(email, password) – passes email, password to backend
- login(req, res) – compares the password with the stored hashed password, and generates and sets jwt token if matched.
- loginWithGoogle(credentials) – passes google credentials to backend
- googleAuth(req, res) – verifies the crendential with google client id and get payload, if payload exist generate and set jwt token else create the user on db and generate and set jwt token.

Image Compression Module:

The image compression module contains image compressor that run solely on the client-side.

- compressImage(original, quality) – compress image based on that parameters and returns blob, url, size.
- onCompress(id, compressedObj, compressedBlob, originalUrl, originalSize) – passes props to the parent component ImageCompressor.
- uploadToBackend() – sends the file data using FormData().
- handleDownload() – creates ObjectURL of compressedBlob and downloads the file.

PDF Compression Module:

The pdf compression module include pdf compressor that runs on the server-side and uses GhostScript.

- compressPdf(original, quality) – takes the file path and desired quality preset and runs ghostscript with optimized settings based on the quality level.

- Returns object containing, blob, url, obj, size, and output path.
- sendCompressedToClient(res, result, download) – sends compressed pdf back to the frontend with download set to true or false.
- handleDownload() – takes blob and triggers instant browser download

4.2 Testing

Once the implementation phase was complete, the File Compression System was subjected to comprehensive testing to confirm that all modules functioned correctly, efficiently, and fully complied with the specified requirements. Testing represents a critical stage in software development, enabling the identification and correction of logical, design, and functional defects prior to deployment. This process guarantees that the delivered system is reliable, secure, and able to perform all intended operations accurately. The tests performed are described in the following section:

4.2.1 Test Cases for Unit Testing

Table 4.1: User Login using Valid Data

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check User login using valid data	Username: prabhat Password: 123	Logged into the home page	As expected	Pass

Table 4.2: User Login using Invalid Data

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check User login using invalid data	Username = neer Password = test	Display Error Message	As expected	Pass

Table 4.3: User Register with full details

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	User enter full details	Name = Prabhat Gurung Username = prabahtgrg Password = 123	Logged into the dashboard page	As expected	Pass

Table 4.4: User Register with incomplete details

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	User enter incomplete details	Name = Prabhat Gurung Username: Password: 123	Display Error Message	As expected	Pass

Table 4.5: Image Compression

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Upload the Image file for compression	Image Files	Reduce Image size	As expected	Pass

Table 4.6: Pdf Compression

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Upload the Pdf file for compression	Pdf Files	Reduce PDF size	As expected	Pass

4.2.2 Test Cases for System Testing

Table 4.7: Authentication Testing

ID	Test Case Description	Test Steps/Data	Expected Result	Actual Result	Pass/ Fail
1	Authentication module testing	Name: Prabhat Gurung Email: prabhatgurung34@gmail.com Password: 123	Registration success => Redirect to dashboard Login success => Redirect to dashboard	As expected	Pass

Table 4.8: Compression Testing

ID	Test Case Description	Test Step	Expected Result	Actual Result	Pass/Fail
1	Image and pdf compression	Upload image and pdf files	On Image upload compressed image generated On Pdf upload compressed Pdf file generated	As expected	Pass

Table 4.9: Download and Share Testing

ID	Test Case Description	Test Step	Expected Result	Actual Result	Pass/Fail
1	Testing Download and Share Functionality	After compression click on the download button	Download and share link generated	As expected	Pass

4.3 Result Analysis

The developed File Compression System underwent comprehensive evaluation through rigorous unit testing, integration testing, and end-to-end system testing. The results conclusively demonstrate that all core components function correctly and reliably under varied test conditions. The user authentication module exhibited robust security and seamless session management. The client-side image compression component consistently delivered high-quality output with configurable quality settings, achieving significant file size reduction while preserving visual fidelity. Similarly, the server-side PDF compression module, powered by Ghostscript successfully processed PDF documents on the selected preset. Integration between frontend and backend components was verified to be stable, with proper handling of file uploads, processing feedback, and secure delivery of compressed assets. The system accurately reports compression metrics (original size, compressed size, and percentage reduction), enhancing user transparency and trust. Overall, the evaluation confirms that the system fully satisfies its primary functional and non-functional requirements, providing an efficient, secure, and user-friendly solution for both image and PDF file optimization within a unified architecture.

Chapter 5: Conclusion and Future Recommendation

5.1 Conclusion

The File Compression System successfully addresses the need for an accessible, efficient, and user-friendly platform for reducing file sizes while maintaining acceptable quality. By leveraging modern web technologies, the system provides a robust solution suitable for both casual users and professionals who require consistent compression capabilities. Its architecture ensures reliability and scalability while integrating key functionalities such as image and PDF compression.

Client-side image compression, implemented through the browser-image-compression library, demonstrates the benefits of using browser capabilities to reduce server load and provide immediate feedback. Features like the quality slider and before-after comparison tools allow users to make informed decisions about the quality-size tradeoff, overcoming limitations of traditional compression tools that lack transparency. This approach improves both speed and overall user experience compared to server-only solutions.

Server-side PDF compression complements the image compression functionality, offering comprehensive file optimization within a single platform. The dual-tier access model balances ease of use for guest users with enhanced features for authenticated users, including compression history and file sharing with expiration and download limits. These features not only incentivize registration but also enable collaborative use and better file organization, addressing common gaps in existing tools.

Overall, the system meets its objectives by providing accessible compression, transparent quality control, secure authentication, history tracking, and file sharing. Performance benchmarks indicate efficient operation, with image compression completing promptly and database operations supporting projected user loads. The File Compression System demonstrates how thoughtful technology choices and algorithms can create practical, user-friendly tools and serves as a foundation for future enhancements or wider deployment.

5.2 Future recommendation

While the current implementation successfully fulfills the core requirements, several enhancements could expand the system's capabilities and improve user experience:

- Extending compression capabilities to support other file formats.

- Integration of machine-learning based compression techniques could achieve superior compression ratio.
- Direct integration with cloud storage platforms including Google Drive, Dropbox, OneDrive, and Amazon S3.
- Cron jobs to automate file deletion after a certain period of time.

References

[1] J. A. &. J. Kari, Digital Image Compression.

[2] M. A. R. a. M. Hamada, "Lossless Image Compression Techniques: A State-of-the-Art Survey," 2019.

[3] M. A. A.-j. a. S. Y. Hamid, "Image Compression Techniques: Literature Review," 2021.

[4] R. Hatlapatka and P. Sojka, "PDF Enhancements Tools for a Digital Library".

[5] D. P. Armando J. Pinho, "MFCompress: a compression tool for FASTA and multi-FASTA data," *Bioinformatics,* vol. 30, no. 1.

[6] A. E. A. &. M. Arshad, "MZPAQ: a FASTQ data compression tool," vol. 14, 2019.

Appendices

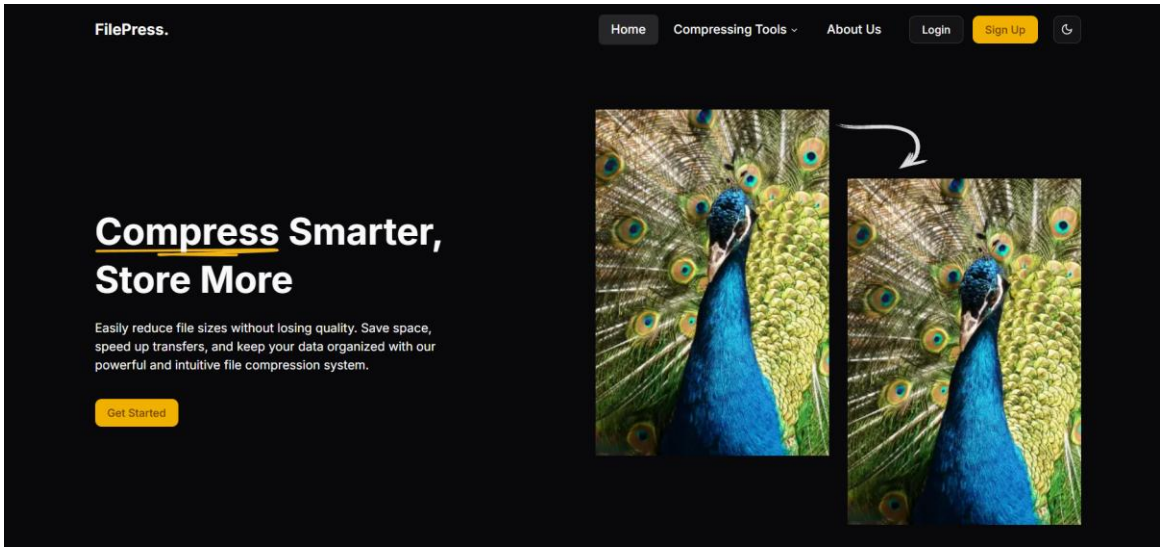

Figure 1: Home Page

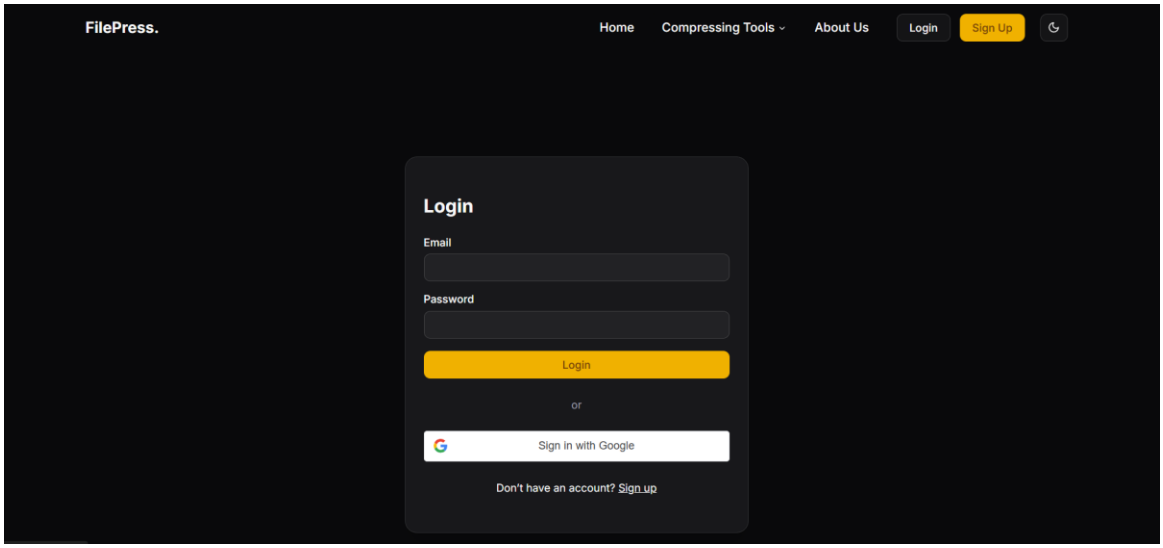

Figure 2: Login Page

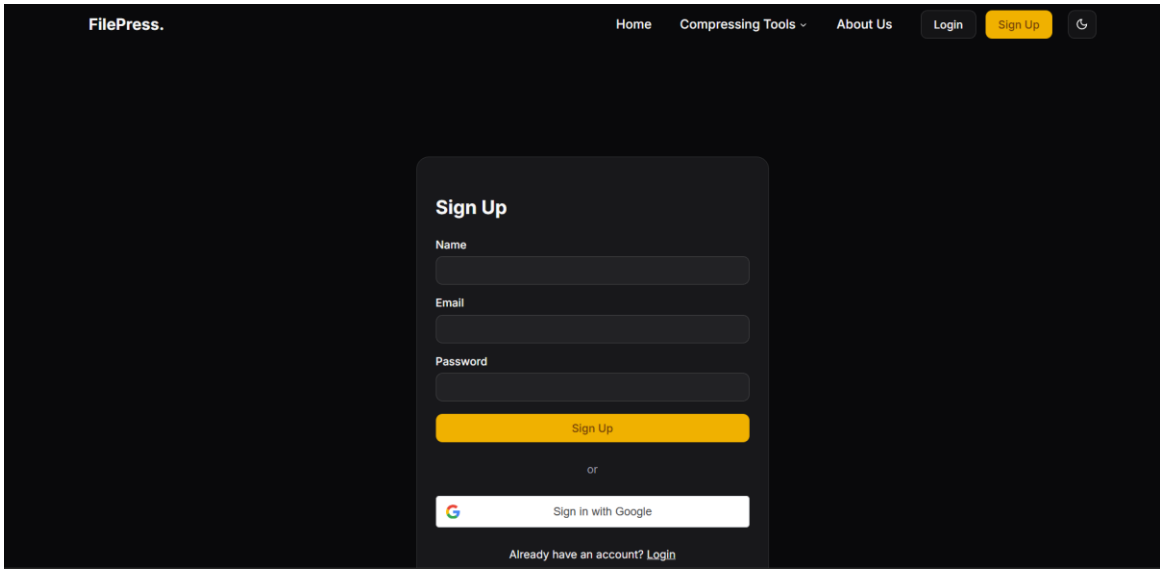

Figure 3: Sign Up Page

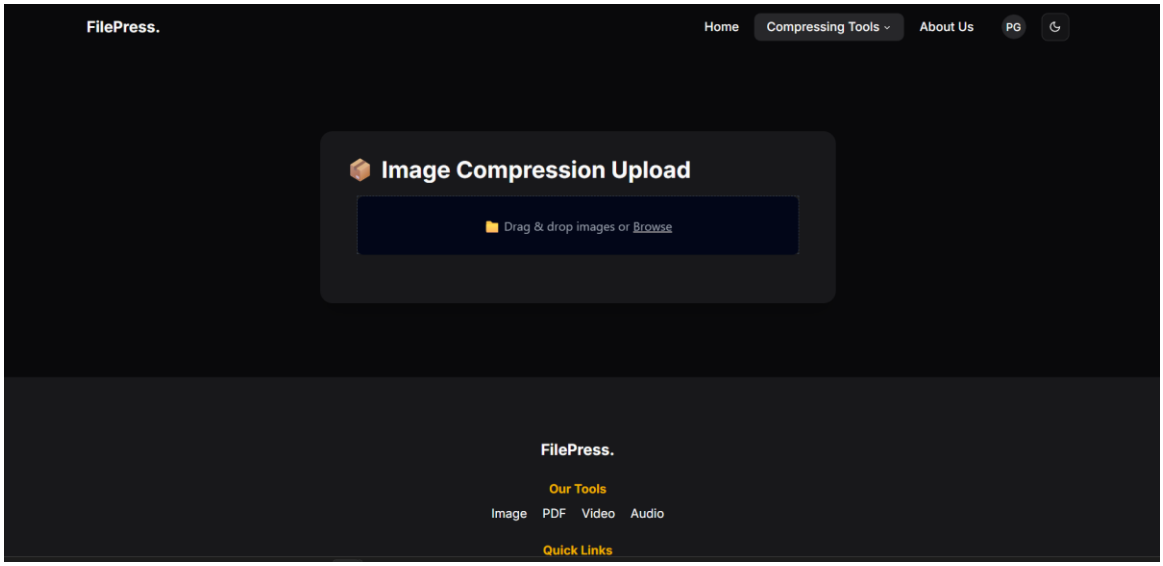

Figure 4: Image Compression Uploader

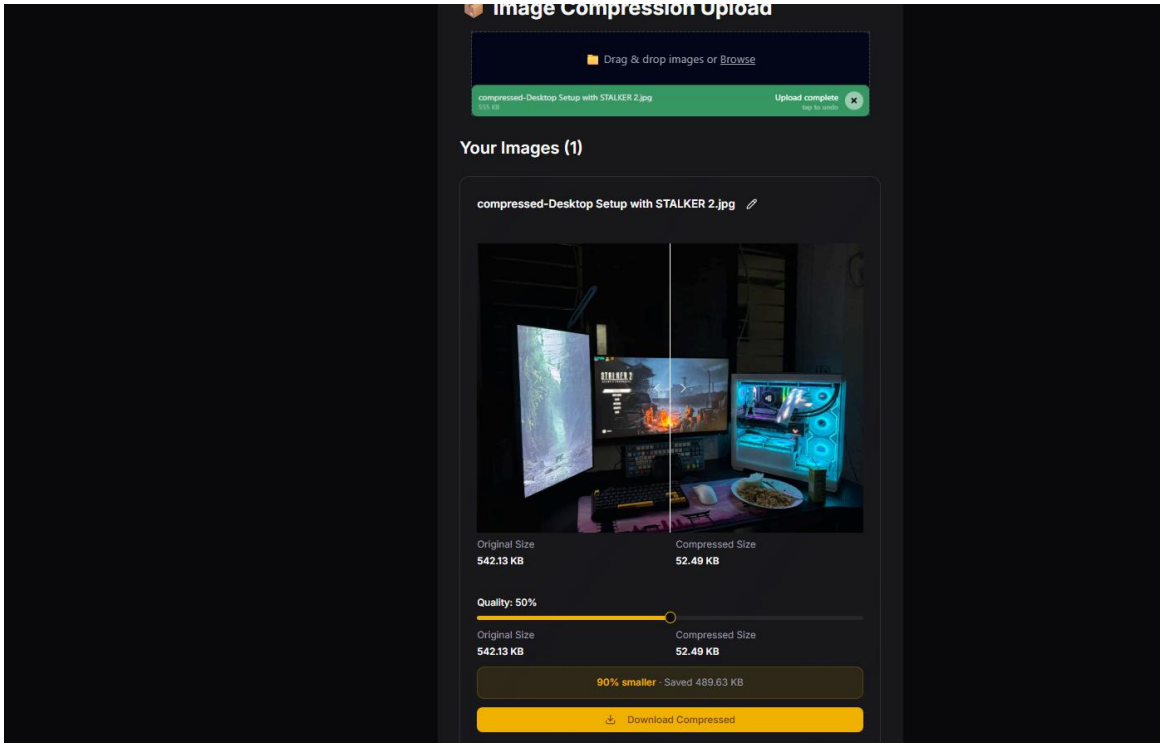

Figure 5: Image Quality and Comparison Slider

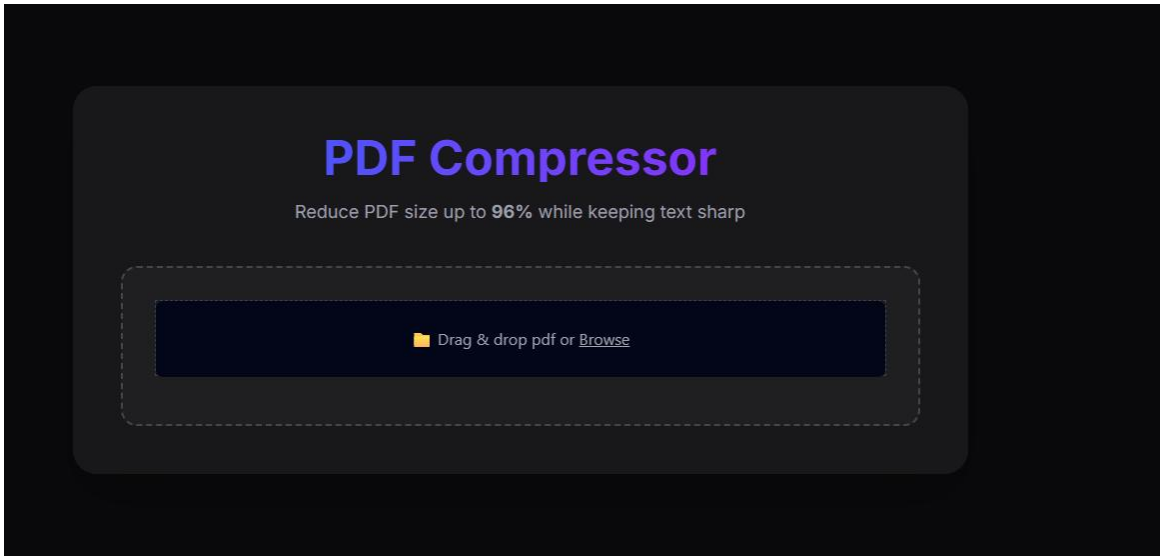

Figure 6: PDF Compressor

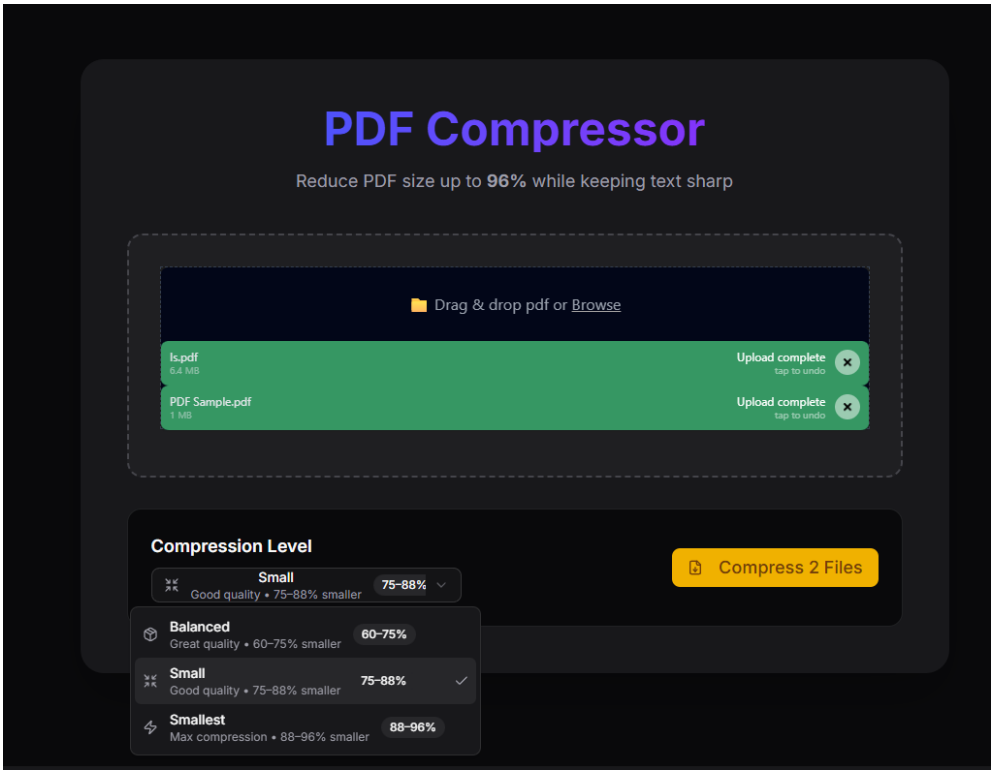

Figure 7: PDF Quality Presets

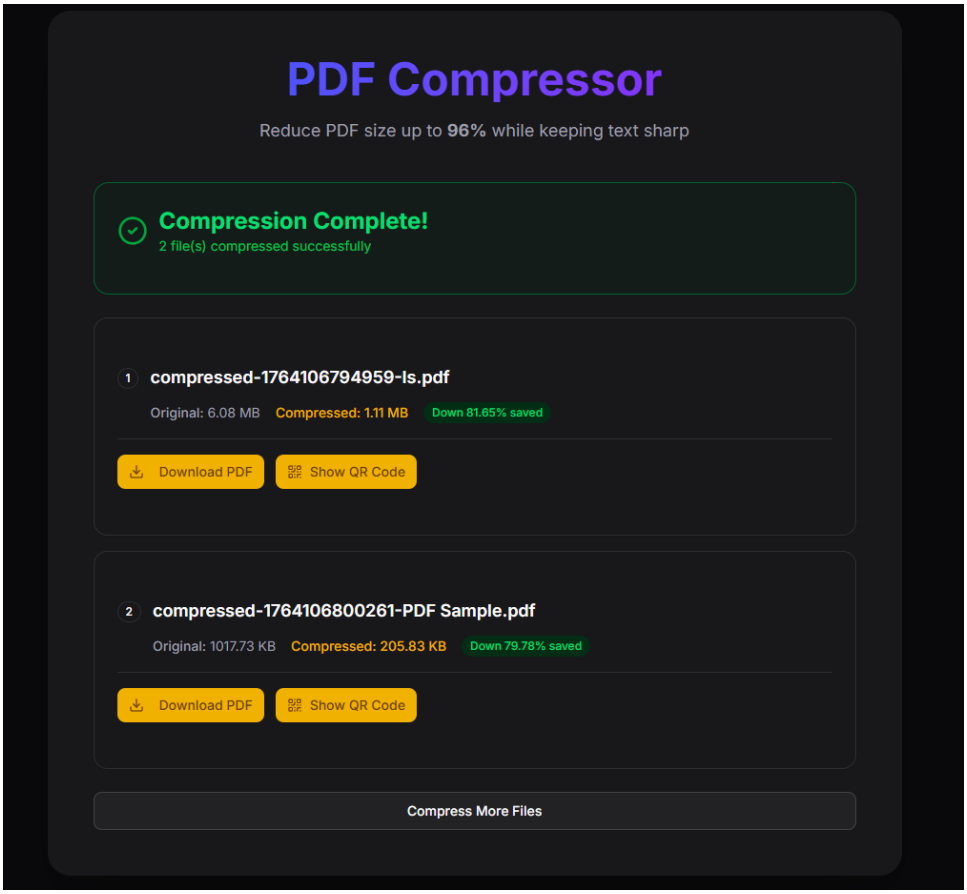

Figure 8: PDF Compression Complete

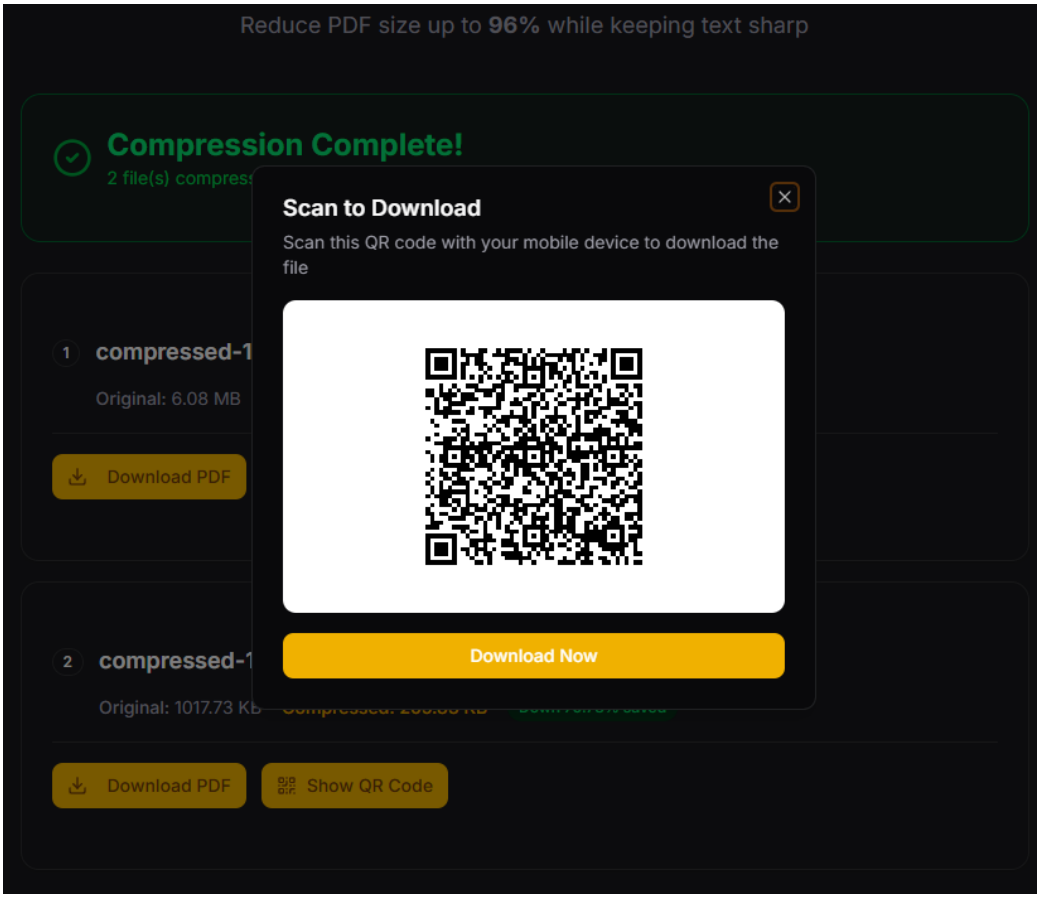

Figure 9: Compressed File Sharing QR

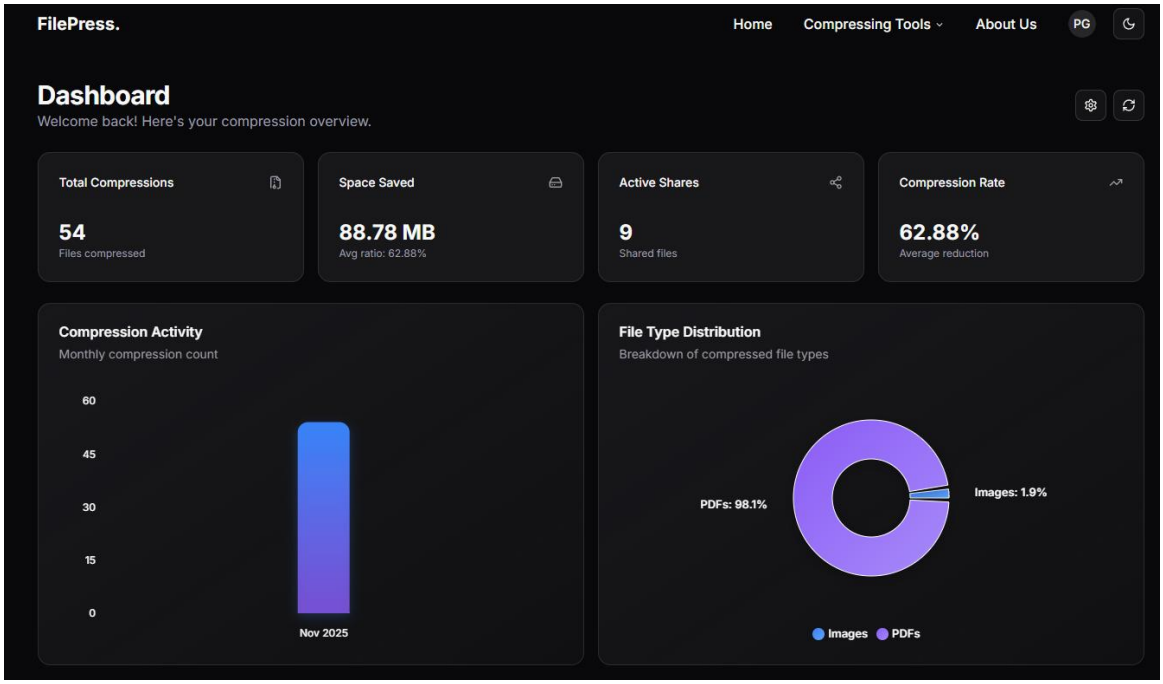

Figure 10: FilePres Dashboard

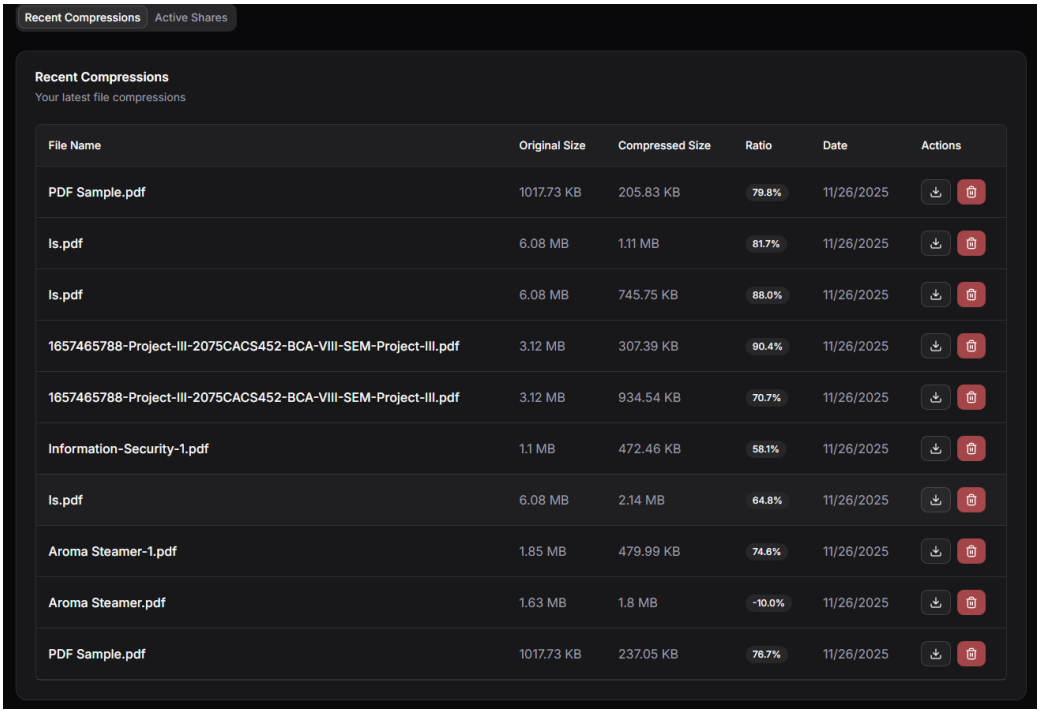

Figure 11: Recent Compressions

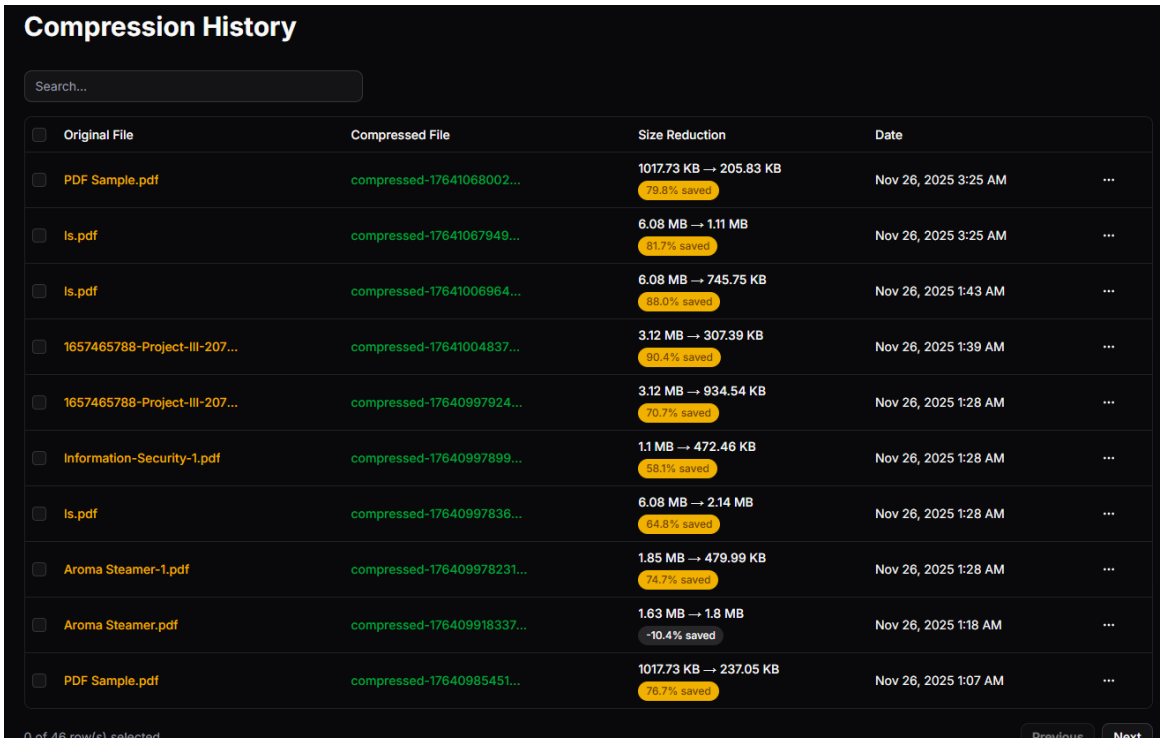

Figure 12: Compression History