# **Compression Portal**

This web-based application allows users to compress and decompress files using various lossless algorithms. It showcases custom implementations of Run-Length Encoding (RLE), Huffman Coding, and LZ77, along with the industry-standard Gzip algorithm (powered by Node.js's zlib). This blend enables users to experiment with educational algorithms and benchmark them against a production-grade solution, offering an intuitive and engaging experience.

## **Features**

* **Comprehensive File Handling:** Upload any file type (text, images, PDFs, binary data) from your local system. After processing, both compressed and decompressed files can be seamlessly downloaded, ensuring a complete workflow.
* **Diverse Compression Algorithm Selection:**
  + **Run-Length Encoding (RLE):** Simple, effective for data with long, identical byte sequences. It compresses by storing the repeating byte and its count; effectiveness depends on data redundancy.
  + **Huffman Coding:** A statistical, variable-length coding scheme that assigns shorter codes to more frequent characters. Ideal for text files with uneven character frequencies, like natural language prose.
  + **LZ77 (Lempel-Ziv 77):** A powerful dictionary-based method that replaces repeated data sequences with concise pointers (offset, length). Our implementation uses a hashing dictionary for optimized search, making it effective for recurring phrases or patterns.
  + **Gzip:** Powered by Node.js's zlib module, Gzip is a robust and universally compatible compression method. It internally uses the DEFLATE algorithm (a hybrid of LZ77 and Huffman), achieving highly efficient and consistent compression across diverse data types, often delivering substantial size reductions.
* **Detailed Compression Metrics:** The application provides valuable feedback for each operation, including original file size, compressed size, compression ratio (efficiency indicator), and processing time. These metrics are crucial for understanding each algorithm's real-world performance.

## **Tech Stack Used**

The Compression Portal is built using a modern JavaScript ecosystem for both frontend and backend, ensuring fluid and efficient performance.

* **Frontend Technologies:**
  + **React.js:** For building dynamic and efficient user interfaces.
  + **Tailwind CSS:** A utility-first framework for rapid, responsive UI development.
  + **Axios:** A promise-based HTTP client for seamless API requests.
* **Backend Technologies:**
  + **Node.js:** JavaScript runtime for server-side execution.
  + **Express.js:** Web framework for defining API endpoints.
  + **Multer:** Middleware for handling file uploads (multipart/form-data).
  + **path module:** For cross-platform file path manipulation.
  + **fs module:** For direct file system interactions (reading/writing files, creating directories).
  + **zlib module:** Provides Gzip compression and decompression functionalities.

## **Setup Instructions to Run the Project Locally**

Follow these steps to set up and run the Compression Portal on your local machine.

### **Prerequisites**

* **Node.js:** (LTS version recommended)
* **npm:** (Node Package Manager, installed with Node.js)

### **1. Backend Setup**

1. **Navigate to the backend directory:**  
   cd C:\Users\ponna\OneDrive\Documents\com\compression-portal\backend\  
     
   (Adjust path to your project's location.)
2. **Install dependencies:**  
   npm install
3. **Start the backend server:**  
   npm start  
     
   The server runs on http://localhost:5000.

### **2. Frontend Setup**

1. **Open a new terminal and navigate to the frontend directory:**  
   cd C:\Users\ponna\OneDrive\Documents\com\compression-portal\frontend\  
     
   (Adjust path to your project's location.)
2. **Install dependencies:**  
   npm install
3. **Start the frontend development server:**  
   npm start  
     
   The application opens in your browser at http://localhost:3000.

### **3. Usage**

1. **Access the application:** Go to http://localhost:3000 in your browser.
2. **Upload a file:** Select a file from your computer.
3. **Select an algorithm:** Choose RLE, Huffman, LZ77, or Gzip.
4. **Perform Compression/Decompression:** Click "Compress File" or "Decompress File" (for files previously compressed by this portal).
5. **Review and Download:** View results and download the processed file.

**Important Notes for Testing and Understanding Performance:**

* **Compatibility of Custom Algorithms:** Files compressed by custom RLE, Huffman, or LZ77 (\_rle\_compressed, \_huf\_compressed, \_lz77\_compressed files) are in a proprietary format. They *must* be decompressed by this portal using the same algorithm. Standard OS tools cannot open them.
* **Performance with Image Files:** General-purpose lossless algorithms (RLE, Huffman, LZ77) often perform poorly on typical image formats (JPEG, PNG). JPEGs are inherently lossy-compressed, leaving little redundancy. PNGs use advanced lossless methods. Expect size increases or minimal changes when using custom algorithms on images. Gzip (via Node.js zlib) generally provides broader effectiveness for diverse data, including some images.
* **Clear File Naming Conventions:** Compressed files are named original\_name\_algorithm\_compressed.original\_extension (e.g., my\_document\_lz77\_compressed.txt). Gzip files are original\_name\_gz.original\_extension. Decompressed files typically use \_decompressed in their name.