COMPRESSION TOOLKIT OVERVIEW AND GUIDE

INTRODUCTION

The Compression Toolkit application is a sophisticated software solution designed to streamline the process of managing file sizes and optimizing storage efficiency. Its primary purpose is to offer users a robust set of tools for compressing, decompressing, and managing various file formats without compromising the quality or integrity of the data. As digital files continue to grow in size and complexity, the necessity for effective compression methods has become increasingly important, making the Compression Toolkit an invaluable resource for both individual users and organizations.

One of the standout features of the Compression Toolkit is its user-friendly interface, which allows users to quickly navigate through the various functionalities with ease. Users can compress files in multiple formats, such as ZIP, RAR, and 7z, providing flexibility in file management. Moreover, the application supports batch processing, enabling users to compress or decompress multiple files simultaneously, significantly saving time and effort.

FEATURES

The Compression Toolkit boasts a range of key features designed to enhance the user experience while delivering efficient file compression and management. One of the primary functionalities is the file selection process, which simplifies the task of choosing files for compression. Users can easily drag and drop files into the application or browse their system to select multiple files or entire folders at once. This intuitive selection method is coupled with support for various file formats, ensuring that users can work with their preferred types without limitation.

INSTALLATION INSTRUCTIONS

To install the Compression Toolkit application, follow these step-by-step instructions. Ensure you meet the prerequisites and execute the specified commands in your terminal.

PREREQUISITES

Before installing the Compression Toolkit, ensure that you have the following software installed on your system:

Node.js: The Compression Toolkit is built on Node.js. You can download the latest version from the official Node.js website. Follow the installation instructions relevant to your operating system (Windows, macOS, or Linux).

npm: Node.js comes with npm (Node Package Manager), which is necessary for installing the Compression Toolkit. Verify that npm is installed by running the following command in your terminal:

INSTALLATION STEPS

Open Terminal: Access your command line interface (Terminal on macOS/Linux or Command Prompt/PowerShell on Windows).

Clone the Repository: Use the following command to clone the Compression Toolkit repository from GitHub:

Replace `your-repo` with the actual repository name if different.

Navigate to the Directory: Change your working directory to the cloned repository:

Install Dependencies: Run the following command to install all the required dependencies:

Configuration Settings: If the application requires any configuration settings, create a configuration file named `.env` in the root directory. You can set environment variables such as:

Adjust the values according to your preferences.

Start the Application: Finally, launch the Compression Toolkit application with the following command:

You should see the application running in your terminal. Follow any additional prompts or instructions provided by the application interface.

CODE STRUCTURE

The organization of the Compression Toolkit codebase is designed for modularity and maintainability, featuring a clear separation of concerns among its main components. This architecture enhances collaboration among developers and facilitates easier updates and debugging. The code is below

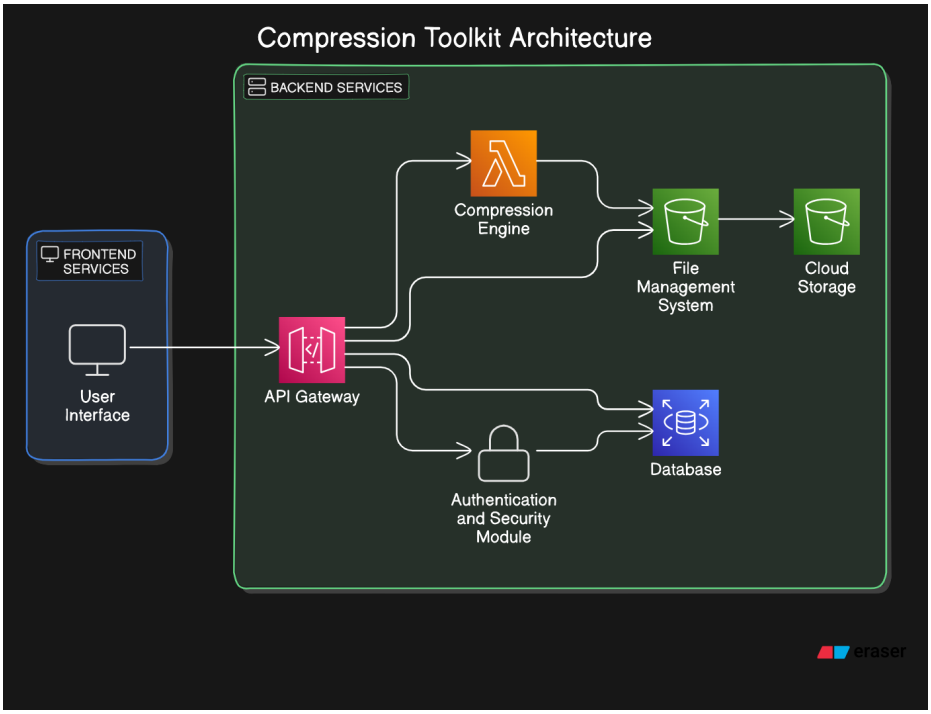
```
import { useState } from 'react';
import { Button } from "/components/ui/button";
import { Input } from "/components/ui/input";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "/
components/ui/select";

function App() {
 const [selectedFiles, setSelectedFiles] = useState([]);
 const [compressionLevel, setCompressionLevel] = useState(50);
 const [compressedFiles, setCompressedFiles] = useState([]);
 const [algorithm, setAlgorithm] = useState('lossless');

 const handleFileChange = (event) => {
   setSelectedFiles(event.target.files);
 };

 const handleCompressionChange = (event) => {
   setCompressionLevel(event.target.value);
 };

 const handleCompress = () => {
   const compressed = [];
   for (const file of selectedFiles) {
     const reader = new FileReader();
     reader.onload = () => {
       const compressedFile = compressFile(reader.result, compressionLevel);
       compressed.push(compressedFile);
       setCompressedFiles(compressed);
     };
     reader.readAsArrayBuffer(file);
   }
 };
```

```
 const compressFile = (file, compressionLevel) => {
   // TO DO: implement compression algorithm
   return file;
 };

 const downloadCompressedFiles = () => {
   for (const file of compressedFiles) {
     const blob = new Blob([file], { type: 'application/octet-stream' });
     const url = URL.createObjectURL(blob);
     const a = document.createElement('a');
     a.href = url;
     a.download = 'compressed_file';
     a.click();
   }
 };

 const handleAlgorithmChange = (value) => {
   setAlgorithm(value);
 };

 return (
   <div className="flex flex-col items-center justify-center h-screen bg-
indigo-500">
     <h1 className="text-5xl font-bold text-white mb-8">Compression Toolkit</
h1>
     <input type="file" multiple onChange={handleFileChange}
className="text-lg font-bold text-white bg-indigo-700 p-2 rounded-lg" />
     <div className="flex items-center space-x-2 mt-8">
       <span className="text-lg font-bold text-white">Compression Level:</
span>
       <input type="range" min="1" max="100" value={compressionLevel}
onChange={handleCompressionChange} className="w-64 h-2 bg-indigo-700
rounded-lg" />
       <span className="text-lg font-bold text-white">{compressionLevel}%</
span>
     </div>
     <div className="flex items-center space-x-4 mt-8">
       <Button variant="primary" className="bg-orange-500 hover:bg-
orange-700 text-white font-bold py-2 px-4 rounded-lg"
onClick={handleCompress}>Compress</Button>
       <Button variant="secondary" className="bg-teal-500 hover:bg-teal-700
text-white font-bold py-2 px-4 rounded-lg"
```

```
onClick={downloadCompressedFiles}>Download Compressed Files</Button>
    </div>
    <div className="flex items-center space-x-2 mt-8">
      <Select>
        <SelectTrigger className="w-[180px] bg-indigo-700 text-white font-bold
py-2 px-4 rounded-lg">
          <SelectValue placeholder="Select compression algorithm" />
        </SelectTrigger>
        <SelectContent>
          <SelectItem value="lossless" onClick={() =>
handleAlgorithmChange('lossless')}>Lossless</SelectItem>
          <SelectItem value="lossy" onClick={() =>
handleAlgorithmChange('lossy')}>Lossy</SelectItem>
        </SelectContent>
      </Select>
      <span className="text-lg font-bold text-white">Selected Algorithm:
{algorithm}</span>
    </div>
  </div>
 );
}

export default App;
```

ARCHITECTURAL DESIGN DIAGRAM

Below is the architectural diagram for the Compression Toolkit, illustrating the high-level design and interactions between its core components, databases, and servers.



COMPONENTS OVERVIEW

User Interface (UI): The entry point for users, the UI allows for intuitive interactions with the Compression Toolkit. It is built using modern web technologies and communicates with the backend services through RESTful APIs. Users can upload files, select compression options, and receive feedback on their actions.

API Gateway: This component serves as the intermediary between the UI and the backend services. It handles incoming requests from the UI, routes them to the appropriate services, and returns responses. The API Gateway ensures security and scalability, managing user authentication and rate limiting.

Compression Engine: The heart of the toolkit, the Compression Engine executes the compression algorithms on the uploaded files. It receives data from the API Gateway, processes the files based on user-defined settings, and returns the results back to the API Gateway for user access.

File Management System: This module manages all file operations, including reading, writing, and storing files. It handles the temporary files generated during the compression process and ensures that the final compressed files are stored securely. This system also interacts with cloud storage services for enhanced accessibility.

Database: The database stores user profiles, compression settings, and logs of compression activities. It supports efficient querying and data retrieval,

enabling the application to maintain user preferences and provide analytics on compression performance.

Authentication and Security Module: This component is responsible for managing user authentication, session control, and file encryption. It ensures that user data is secure while being processed and stored, adhering to best practices in data protection.

Cloud Storage: For users requiring additional space or remote access, the toolkit integrates with cloud storage services. This allows users to easily upload their compressed files to a secure environment, making them accessible from anywhere.

USAGE INSTRUCTIONS

Once you have successfully installed the Compression Toolkit application, operating it is straightforward and user-friendly. This section provides a detailed guide on how to navigate the application, upload files, select compression levels, initiate the compression process, and download your compressed files.

UPLOADING FILES

To begin using the Compression Toolkit, you need to upload the files you wish to compress. You can do this in two ways:

Drag and Drop: Simply drag the files or folders you want to compress and drop them into the designated area of the application interface. The application will automatically recognize the files for processing.

File Selection: Alternatively, you can click on the "Upload" button within the application. This action will open a file explorer window where you can browse your system. Select the desired files or folders and click "Open" to upload them into the toolkit.

SELECTING COMPRESSION LEVELS

Once the files are uploaded, you will need to select the desired compression level. The Compression Toolkit offers various options ranging from low to high compression:

- **Low Compression**: Prioritizes speed and is suitable for users who need quick processing times.
- **Medium Compression**: A balanced choice that offers a moderate reduction in file size without significantly affecting speed.
- **High Compression**: Maximizes space savings, ideal for users who want to reduce file sizes as much as possible, even if it takes a little longer.

Select your preferred option from the dropdown menu or slider available in the interface.

INITIATING COMPRESSION

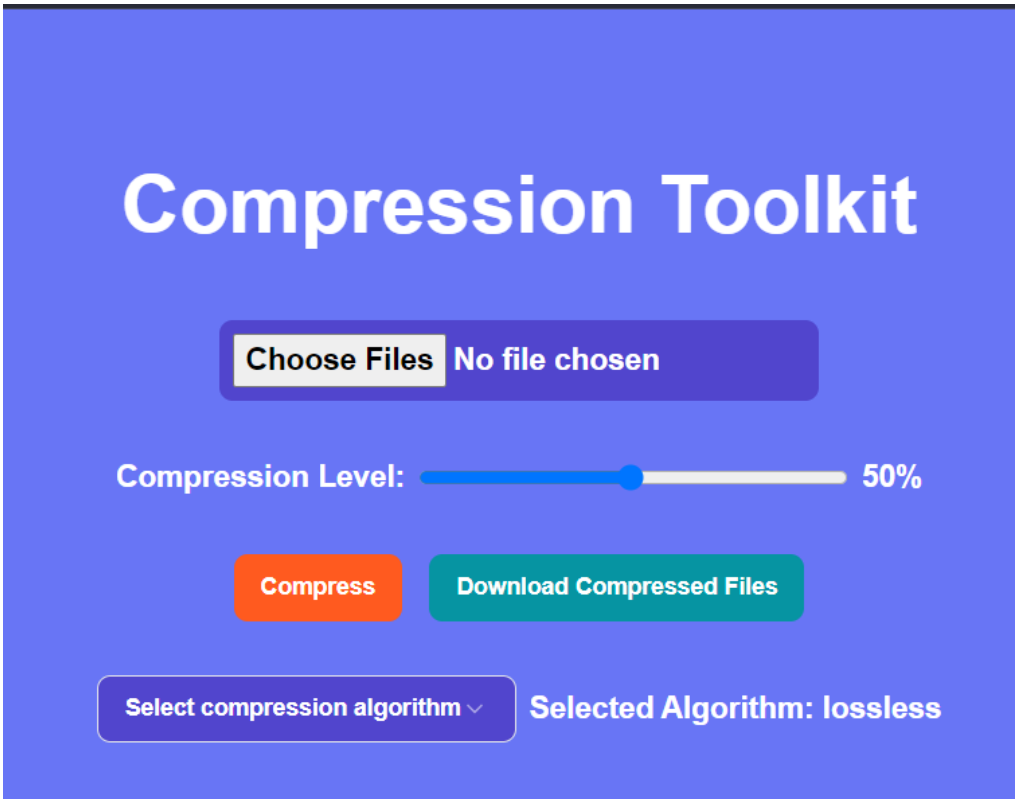
After selecting the compression level, click on the "Compress" button to start the compression process. The application will display a progress indicator, allowing you to monitor the status. Depending on the size of the files and the selected compression level, this process may take some time.

DOWNLOADING FILES

Once the compression is complete, the application will notify you and provide options for downloading your compressed files. You can click on the "Download" button to save the compressed files directly to your device. Additionally, the toolkit may also offer options to upload the compressed files to cloud storage or share them via email, enhancing your flexibility in managing files.

With these steps, you can efficiently use the Compression Toolkit to manage your files, ensuring they are optimized for storage without sacrificing quality.

Preview of the App




```
npm start
```

```
COMPRESSION_LEVEL=medium
ENCRYPTION_KEY=your_secret_key
```

```
npm install
```

```
cd compression-toolkit
```

```
git clone https://github.com/your-repo/compression-
toolkit.git
```

```
npm -v
```