File Upload and Management System Documentation

By Shadman Sakib

Table of Contents

Introduction

Introducing the File Upload and Management API: a handy tool for uploading, downloading, and organizing your files. It's user-friendly, with features for uploading different file types. When you upload a file, you get two keys – a public key to share files and a private key to control and delete them.

To keep things fair, the API has limits. You can upload a maximum of 6 files at a time. Additionally, you are allowed to upload files up to 10 times a day and download files up to 10 times a day. Plus, it takes care of cleanup by automatically deleting files that haven't been touched for about 7 minutes.

This API is your go-to solution for easy and secure file management, making sharing, downloading, and cleaning up files a breeze. Welcome to a better way of handling your files with the File Upload and Management API!

Features

- File Upload
- File Download
- File Deletion
- Rate Limiting for Upload and Download
- Automatic File Cleanup

Prerequisites

Before using the API, ensure you have the following prerequisites:

- Node.js installed on your server
- NPM (Node Package Manager) for installing dependencies
- A server or hosting environment to run the API

Getting Started

- Clone the project repository to your server.
- Install project dependencies by running npm install.
- Start the API by running npm start.
- The API will be accessible at http://your-server-url:3500.
- Make sure to specify the folder path where you want to store files in the environment (env) configuration using the variable named FOLDER.

API Routes

Upload Files

- Route: POST /upload
- Description: Upload one or multiple files to the server. Uploaded file is associated with a unique public and private key.
- Rate Limit: 10 requests per day for uploads, with a maximum of 6 files per request.
- Request: Multipart form-data with files.
- Response: Upon successful upload, the API will return a JSON response with the status of the upload, along with a single pair of public and private keys.

Download Files

- Route: GET /download/:publicKey
- Description: Download files associated with a specific public key. The files will be compressed in a zip archive for download.
- Rate Limit: 10 requests per day for downloads, with a maximum of 6 files per request.
- Request: The public key in the route parameter.
- Response: A zip archive containing the requested files.

Delete Files

- Route: GET /delete/:privateKey
- Description: Delete files associated with a specific private key.
- Request: The private key in the route parameter.
- Response: JSON response indicating the status of the file deletion.

Rate Limiting

The API implements rate limiting to prevent abuse. Users can upload files up to 10 times a day and download files up to 10 times a day. Rate-limiting error responses will be returned if limits are exceeded.

File Cleanup

The API includes an automatic file cleanup process. Files that have not been accessed or modified for more than 5 days will be automatically deleted. The cleanup task runs daily at midnight (00:00)

Dependencies

- Express.js: A web application framework for Node.js.
- Express-FileUpload: Middleware for handling file uploads.
- Bcrypt: A library for secure password hashing.
- Dotenv: A module for loading environment variables from a .env file.
- Jest: A testing framework for JavaScript applications.
- Node-Cron: A job scheduler for running cleanup tasks.
- Express-Rate-Limit: Middleware for rate limiting requests.
- Nodemon: A utility that automatically restarts your Node.js application when files change.
- Archiver: A library for creating zip archives.
- UUID: A package for generating unique keys.
- Supertest: A testing library for making HTTP assertions.
  .

Middleware

- Name : FileExtLimiter
  Description: The FileExtLimiter middleware is responsible for limiting the types of file extensions that can be uploaded. It ensures that only specified file extensions are allowed while blocking all others. This helps maintain security and restricts the types of files that can be uploaded to the server.

- Name: FileSizeLimiter
  Description: The FileSizeLimiter middleware controls the maximum size of files that can be uploaded to the server. It restricts files that exceed a defined size, preventing excessively large files from overloading the server and affecting performance.

- Name: FilesPayloadExist
  Description: The FilesPayloadExist middleware checks whether the request contains files to upload. It ensures that the request payload includes the necessary file data, preventing unnecessary processing when no files are included in the request.

- Name: rateLimitMiddleware
  Description: The rateLimitMiddleware enforces rate limiting on incoming requests to prevent abuse and ensure fair use of the API. It restricts the number of requests that can be made within a specified time frame, helping to maintain server performance and prevent overuse.