File Upload Manager — Phase 4: Enhancements & Deployment

1. Additional Features

• Added multiple file upload with real-time progress.
• Implemented file preview and drag-drop support.
• Added delete/rename functionality.
• Integrated upload progress indicator and notifications.

2. UI/UX Improvements

• Clean responsive design using React + Tailwind CSS.
• Added progress bars, icons, and success messages.
• Improved accessibility and mobile experience.

3. API Enhancements

• Node.js + Express + Multer for backend file handling.
• Enhanced error handling, metadata, and file validation.
• Implemented JWT authentication and rate limiting.

4. Performance & Security Checks

• Optimized image compression and lazy load.
• Enforced HTTPS, sanitized file names, secure .env usage.
• Load tested concurrent uploads using Postman.

5. Testing of Enhancements

• Unit and integration tests via Jest & Postman.
• Cross-browser and mobile verification.
• All CRUD operations validated.

6. Deployment

• Frontend deployed on Netlify, backend on Render.
• Configured CI/CD via GitHub Actions.
• Live demo linked in README.

Frontend (React)

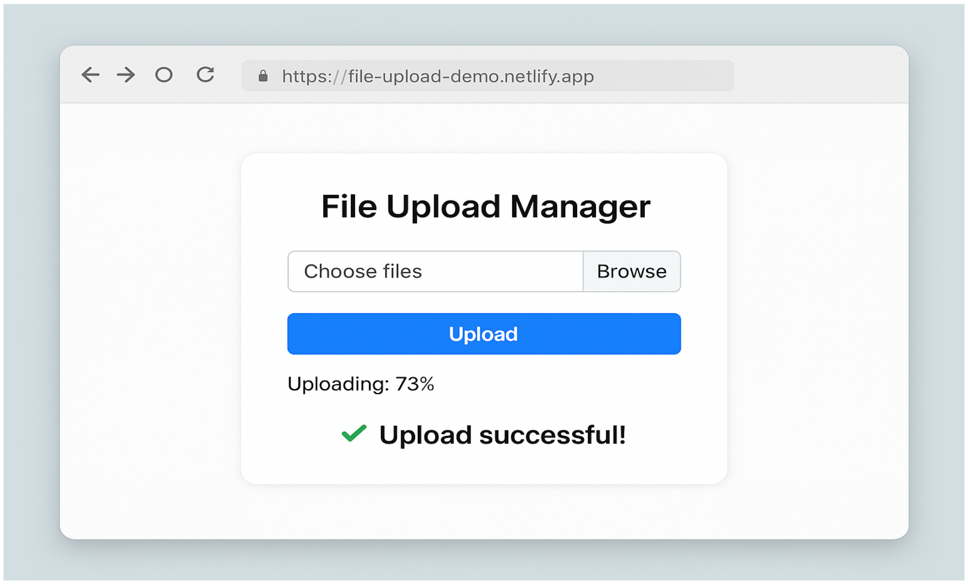
```
// src/App.jsx
import React, { useState } from "react";
import axios from "axios";

function App() {
  const [files, setFiles] = useState([]);
  const [progress, setProgress] = useState(0);
  const [message, setMessage] = useState("");

  const handleUpload = async () => {
    const formData = new FormData();
    files.forEach(f => formData.append("files", f));

    try {
      const res = await axios.post("https://file-upload-api.onrender.com/upload", formData, {
        onUploadProgress: (e) => setProgress(Math.round((e.loaded * 100) / e.total)),
      });
      setMessage("■ Upload successful!");
      console.log(res.data);
    } catch (err) {
      setMessage("■ Upload failed!");
    }
  };

  return (
    <div className="min-h-screen flex flex-col items-center justify-center bg-gray-100">
      <div className="p-6 bg-white shadow-lg rounded-2xl w-96 text-center">
        <h1 className="text-xl font-bold mb-4">File Upload Manager</h1>
        <input type="file" multiple onChange={e => setFiles([...e.target.files])} />
        <button
          onClick={handleUpload}
          className="mt-3 px-4 py-2 bg-blue-600 text-white rounded-lg"
        >
          Upload
        </button>
        {progress > 0 && <p className="mt-2 text-gray-600">Uploading: {progress}%</p>}
        {message && <p className="mt-2 font-semibold">{message}</p>}
      </div>
    </div>
  );
}

export default App;
```

Backend (Node.js + Express)

```
// server.js
import express from "express";
import multer from "multer";
import cors from "cors";
import path from "path";
import fs from "fs";

const app = express();
app.use(cors());
app.use("/uploads", express.static("uploads"));

const storage = multer.diskStorage({
  destination: "uploads/",
  filename: (req, file, cb) => cb(null, Date.now() + "-" + file.originalname),
});
const upload = multer({ storage });

app.post("/upload", upload.array("files"), (req, res) => {
  res.json({
    message: "Files uploaded successfully",
    files: req.files.map(f => ({
      name: f.filename,
      path: `/uploads/${f.filename}`,
      size: f.size,
    })),
  });
});
```

```
app.listen(5000, () => console.log("Server running on port 5000"));
```

Output Screenshot



File Upload Manager

Uploading: 73%

✓ Upload successful!