**Project 2: Image Compression & Analytics App**

**Code:**

import { useState } from "react";

import imageCompression from "browser-image-compression";

import "./ImageCompressor.css";

export default function ImageCompressor() {

  const [selectedImage, setSelectedImage] = useState(null);

  const [compressedImage, setCompressedImage] = useState(null);

  const [compressionStats, setCompressionStats] = useState(null);

  const handleImageUpload = async (event) => {

    const imageFile = event.target.files[0];

    if (!imageFile) return;

    setSelectedImage(imageFile);

    const options = {

      maxSizeMB: 1,

      maxWidthOrHeight: 800,

      useWebWorker: true,

    };

    try {

      const compressedFile = await imageCompression(imageFile, options);

      setCompressedImage(compressedFile);

      setCompressionStats({

        originalSize: (imageFile.size / 1024).toFixed(2),

        compressedSize: (compressedFile.size / 1024).toFixed(2),

        reduction: (

          ((imageFile.size - compressedFile.size) / imageFile.size) \* 100

        ).toFixed(2),

      });

    } catch (error) {

      console.error("Compression error:", error);

    }

  };

  const handleDownload = () => {

    if (compressedImage) {

      const url = URL.createObjectURL(compressedImage);

      const link = document.createElement("a");

      link.href = url;

      link.download = "compressed-image.jpg";

      document.body.appendChild(link);

      link.click();

      document.body.removeChild(link);

    }

  };

  return (

    <div className="container">

      <h1 className="title">Image Compression & Analytics</h1>

      <input type="file" accept="image/\*" onChange={handleImageUpload} className="file-input" />

      {selectedImage && (

        <div className="image-container">

          <p className="label">Original Image:</p>

          <img src={URL.createObjectURL(selectedImage)} alt="Original" className="image" />

        </div>

      )}

      {compressedImage && (

        <div className="image-container">

          <p className="label">Compressed Image:</p>

          <img src={URL.createObjectURL(compressedImage)} alt="Compressed" className="image" />

          <button className="download-button" onClick={handleDownload}>Download Compressed Image</button>

        </div>

      )}

      {compressionStats && (

        <div className="stats-container">

          <p>Original Size: {compressionStats.originalSize} KB</p>

          <p>Compressed Size: {compressionStats.compressedSize} KB</p>

          <p>Size Reduction: {compressionStats.reduction}%</p>

        </div>

      )}

    </div>

  );

}

**Backend:**

const express = require("express");

const multer = require("multer");

const path = require("path");

const fs = require("fs");

const sharp = require("sharp");

const cors = require("cors");

const app = express();

const port = 5000;

// Middleware

app.use(cors());

app.use(express.json());

app.use(express.static("uploads"));

// Set up multer for file uploads

const storage = multer.diskStorage({

  destination: "uploads/",

  filename: (req, file, cb) => {

    cb(null, Date.now() + path.extname(file.originalname));

  },

});

const upload = multer({ storage: storage });

// Image upload and compression endpoint

app.post("/upload", upload.single("image"), async (req, res) => {

  try {

    if (!req.file) {

      return res.status(400).json({ error: "No file uploaded" });

    }

    const inputPath = req.file.path;

    const outputPath = `uploads/compressed-${req.file.filename}`;

    await sharp(inputPath)

      .resize({ width: 800 })

      .jpeg({ quality: 60 })

      .toFile(outputPath);

    // Delete the original file

    fs.unlinkSync(inputPath);

    res.json({

      message: "Image compressed successfully",

      compressedImageUrl: `http://localhost:${port}/${path.basename(outputPath)}`,

    });

  } catch (error) {

    console.error("Error processing image:", error);

    res.status(500).json({ error: "Error compressing image" });

  }

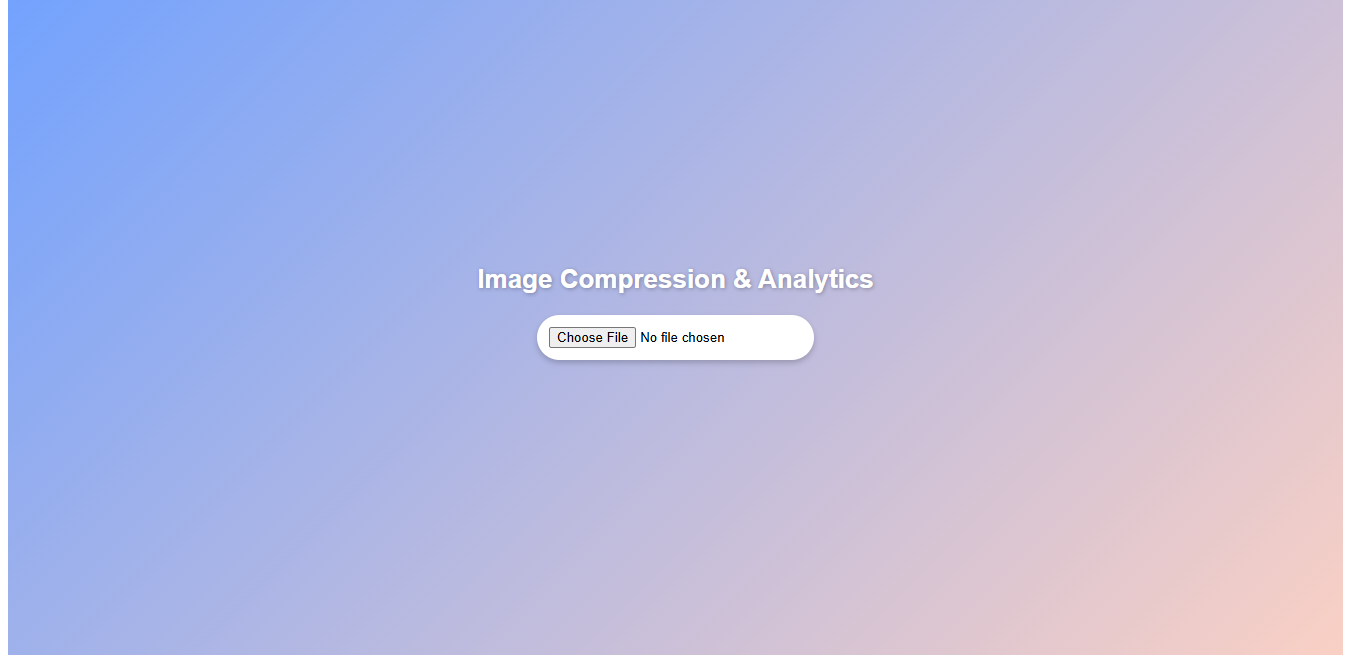
});

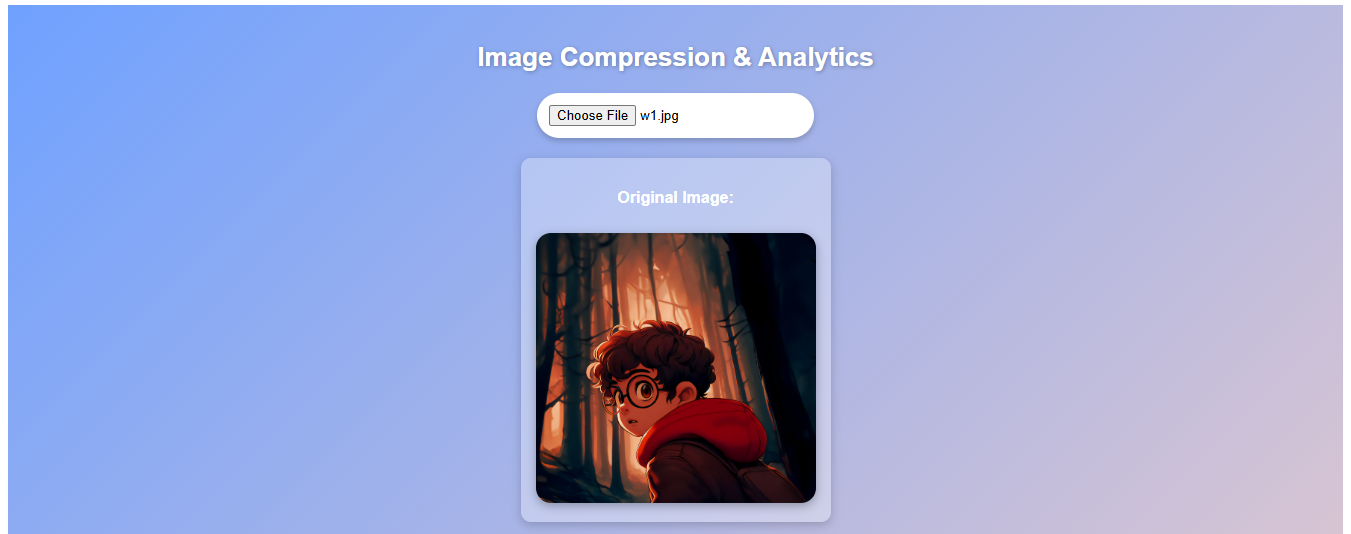
app.listen(port, () => {

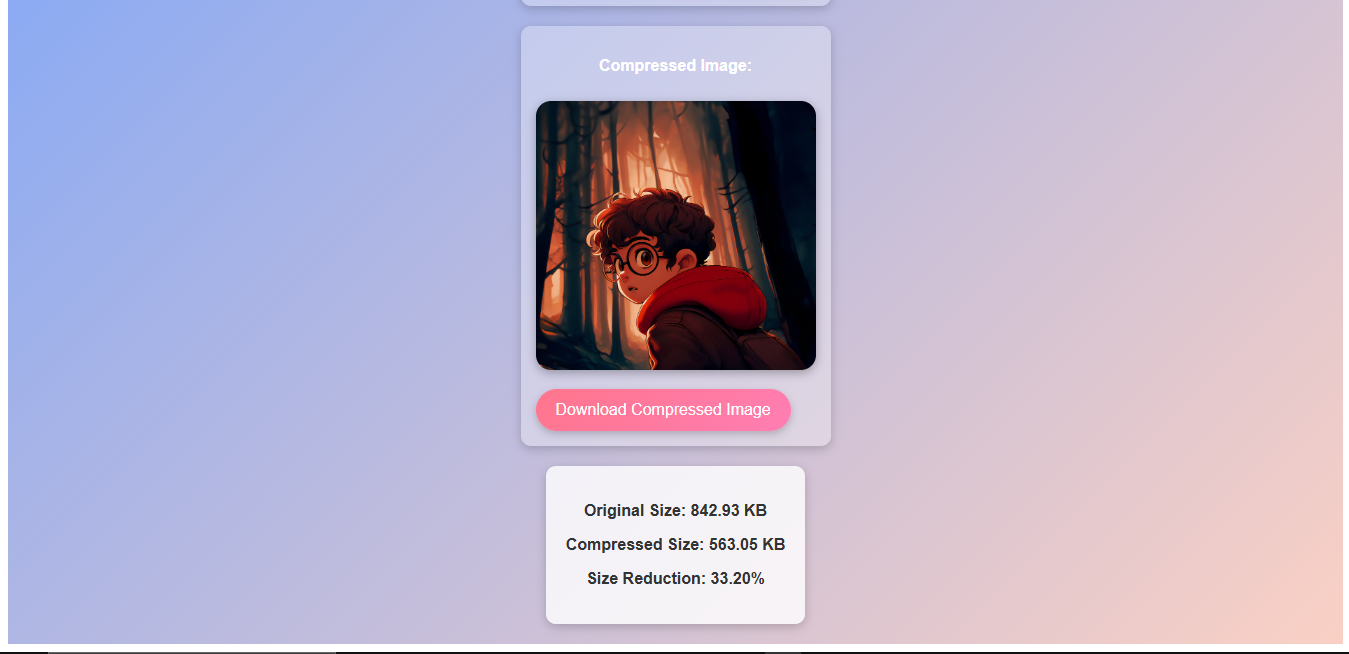
  console.log(`Server is running on http://localhost:${port}`);

});

**Output:**

****

****

****

**Github:**

https://github.com/VandanKambodi/FSWD\_midsem\_exam