Index.js till image upload is successfully in s3+ocr from rekognition--

const express = require('express');

const multer = require('multer');

const AWS = require('aws-sdk');

const { S3Client } = require('@aws-sdk/client-s3');

const { Upload } = require('@aws-sdk/lib-storage');

const { extractTextFromS3Image } = require('./rekognitionUtil');

const { ImageModel } = require('./db');

const cors = require('cors');

require('dotenv').config();

const app = express();

app.use(cors());

*// ✅ Multer config: Store uploaded file in memory*

const storage = multer.memoryStorage();

const upload = multer({ storage });

*// ✅ AWS S3 client setup*

const s3Client = new S3Client({

  region: 'us-east-1',

  credentials: {

    accessKeyId: process.env.AWS\_ACCESS\_KEY\_ID,

    secretAccessKey: process.env.AWS\_SECRET\_ACCESS\_KEY,

  },

});

*// ✅ Upload route with OCR (no classification)*

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

*try* {

    const imageKey = 'uploads/' + req.file.originalname;

    const s3Upload = new Upload({

      client: s3Client,

      params: {

        Bucket: 'productimages2025',

        Key: imageKey,

        Body: req.file.buffer,

        ContentType: req.file.mimetype,

      },

    });

    const result = *await* s3Upload.done();

    console.log('✅ Uploaded to S3 at:', result.Location);

*// OCR*

    const detectedText = *await* extractTextFromS3Image('productimages2025', imageKey);

    console.log('🧠 OCR Text:', detectedText);

*// Save to MongoDB*

    const imageData = new ImageModel({

      filename: imageKey,

      url: result.Location,

      text: detectedText,

    });

*await* imageData.save();

    console.log('📦 Saved to MongoDB');

    res.json({

      message: 'Upload + OCR successful',

      filename: imageKey,

      url: result.Location,

      text: detectedText,

    });

  } *catch* (err) {

    console.error('🔥 Upload/OCR Error:', err);

    res.status(500).json({ error: 'Upload or OCR failed', details: err.message });

  }

});

*// Start your server*

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {*// top of index.js*

  const mongoose = require('mongoose');

  const express = require('express');

  const multer = require('multer');

  const AWS = require('aws-sdk');

  const { S3Client } = require('@aws-sdk/client-s3');

  const { Upload } = require('@aws-sdk/lib-storage');

  const { extractTextFromS3Image } = require('./rekognitionUtil');

  const { ImageModel } = require('./db'); *// Make sure this imports AFTER mongoose connects*

  const cors = require('cors');

  require('dotenv').config();

  const app = express();

  app.use(cors());

*// ✅ Multer config: Store uploaded file in memory*

  const storage = multer.memoryStorage();

  const upload = multer({ storage });

*// ✅ AWS S3 client setup*

  const s3Client = new S3Client({

    region: 'us-east-1',

    credentials: {

      accessKeyId: process.env.AWS\_ACCESS\_KEY\_ID,

      secretAccessKey: process.env.AWS\_SECRET\_ACCESS\_KEY,

    },

  });

*// ✅ Upload route with OCR (no classification)*

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

*try* {

      const imageKey = 'uploads/' + req.file.originalname;

      const s3Upload = new Upload({

        client: s3Client,

        params: {

          Bucket: 'productimages2025',

          Key: imageKey,

          Body: req.file.buffer,

          ContentType: req.file.mimetype,

        },

      });

      const result = *await* s3Upload.done();

      console.log('✅ Uploaded to S3 at:', result.Location);

*// OCR*

      const detectedText = *await* extractTextFromS3Image('productimages2025', imageKey);

      console.log('🧠 OCR Text:', detectedText);

*// Save to MongoDB*

      const imageData = new ImageModel({

        filename: imageKey,

        url: result.Location,

        text: detectedText,

      });

*await* imageData.save();

      console.log('📦 Saved to MongoDB');

      res.json({

        message: 'Upload + OCR successful',

        filename: imageKey,

        url: result.Location,

        text: detectedText,

      });

    } *catch* (err) {

      console.error('🔥 Upload/OCR Error:', err);

      res.status(500).json({ error: 'Upload or OCR failed', details: err.message });

    }

  });

*// ✅ CONNECT TO MONGO & START SERVER*

  const PORT = process.env.PORT || 5001;

  mongoose.connect(process.env.MONGO\_URI, {

    useNewUrlParser: true,

    useUnifiedTopology: true

  }).then(() => {

    console.log("✅ MongoDB connected");

*// ✅ Start server only AFTER DB connects*

    app.listen(PORT, () => {

      console.log(`🚀 Server running on port ${PORT}`);

    });

  }).catch(err => {

    console.error("❌ MongoDB connection error:", err);

  });

  console.log(`🚀 Server running on port ${PORT}`);

});

Uploadimage.js--  
  
*import React, { useState } from 'react';*

*import axios from 'axios';*

*import './ImageUpload.css';*

*function UploadImage() {*

*const [file, setFile] = useState(null);*

*const [previewURL, setPreviewURL] = useState(null);*

*const [uploadMessage, setUploadMessage] = useState('');*

*const [ingredients, setIngredients] = useState([]);*

*const [grade, setGrade] = useState('');*

*const handleFileChange = (e) => {*

*const selected = e.target.files[0];*

*setFile(selected);*

*setPreviewURL(URL.createObjectURL(selected));*

*setUploadMessage('');*

*setIngredients([]);*

*setGrade('');*

*};*

*const handleUpload = async () => {*

*if (!file) {*

*alert("Please select a file first!");*

*return;*

*}*

*const formData = new FormData();*

*formData.append('image', file);*

*try {*

*const res = await axios.post('http://localhost:5000/upload', formData, {*

*headers: { 'Content-Type': 'multipart/form-data' }*

*});*

*const { filename, text, grade } = res.data;*

*setUploadMessage(`✅ Uploaded: ${filename}`);*

*setIngredients(text.split(',').map(str => str.trim()));*

*setGrade(grade || 'N/A');*

*} catch (err) {*

*console.error('❌ Upload failed:', err.response?.data || err.message);*

*setUploadMessage('❌ Upload failed');*

*}*

*};*

*return (*

*<div className="upload-container">*

*<div className="upload-box">*

*<h2>🍽️ Ingredient Grader</h2>*

*<input*

*type="file"*

*accept="image/\*"*

*onChange={handleFileChange}*

*className="upload-input"*

*/>*

*{previewURL && (*

*<div className="preview-container">*

*<h4>Preview:</h4>*

*<img src={previewURL} alt="preview" className="preview-image" />*

*</div>*

*)}*

*<button onClick={handleUpload} className="upload-button">*

*Upload & Analyze*

*</button>*

*{uploadMessage && (*

*<p className={`upload-message ${uploadMessage.startsWith('✅') ? 'upload-message-success' : 'upload-message-error'}`}>*

*{uploadMessage}*

*</p>*

*)}*

*{ingredients.length > 0 && (*

*<div className="ingredient-list">*

*<h3>🧾 Extracted Ingredients:</h3>*

*<ul>*

*{ingredients.map((item, index) => (*

*<li key={index}>{item}</li>*

*))}*

*</ul>*

*<div className="ingredient-grade">*

*🧪 Grade: <span className="grade-badge">{grade}</span>*

*</div>*

*</div>*

*)}*

*</div>*

*</div>*

*);*

*}*

*export default UploadImage;*