// --- Frontend (React.js) ---

// src/components/RecipeForm.js

import React, { useState } from 'react';

const RecipeForm = ({ onAddRecipe, recipeToEdit, onUpdateRecipe }) => {

const [recipe, setRecipe] = useState(recipeToEdit || {

name: '',

ingredients: '',

instructions: '',

category: '',

prepTime: '',

image: null,

tags: '',

});

const handleChange = (e) => {

const { name, value } = e.target;

setRecipe({ ...recipe, [name]: value });

};

const handleImageChange = (e) => {

setRecipe({ ...recipe, image: e.target.files[0] });

};

const handleSubmit = (e) => {

e.preventDefault();

if (recipeToEdit) {

onUpdateRecipe({ ...recipe, \_id: recipeToEdit.\_id });

} else {

onAddRecipe(recipe);

}

setRecipe({

name: '',

ingredients: '',

instructions: '',

category: '',

prepTime: '',

image: null,

tags: '',

});

};

return (

<form onSubmit={handleSubmit}>

<input type="text" name="name" placeholder="Recipe Name" value={recipe.name} onChange={handleChange} required />

<textarea name="ingredients" placeholder="Ingredients" value={recipe.ingredients} onChange={handleChange} required />

<textarea name="instructions" placeholder="Instructions" value={recipe.instructions} onChange={handleChange} required />

<select name="category" value={recipe.category} onChange={handleChange} required>

<option value="">Select Category</option>

<option value="Breakfast">Breakfast</option>

<option value="Lunch">Lunch</option>

<option value="Dinner">Dinner</option>

<option value="Dessert">Dessert</option>

</select>

<input type="text" name="prepTime" placeholder="Preparation Time" value={recipe.prepTime} onChange={handleChange} />

<input type="file" name="image" onChange={handleImageChange} />

<input type="text" name="tags" placeholder="Tags (comma-separated)" value={recipe.tags} onChange={handleChange} />

<button type="submit">{recipeToEdit ? 'Update Recipe' : 'Add Recipe'}</button>

</form>

);

};

export default RecipeForm;

// src/components/RecipeList.js

import React from 'react';

const RecipeList = ({ recipes, onDeleteRecipe, onEditRecipe }) => {

return (

<div>

{recipes.map((recipe) => (

<div key={recipe.\_id}>

<h3>{recipe.name}</h3>

<p>Category: {recipe.category}</p>

<button onClick={() => onDeleteRecipe(recipe.\_id)}>Delete</button>

<button onClick={() => onEditRecipe(recipe)}>Edit</button>

</div>

))}

</div>

);

};

export default RecipeList;

// src/App.js

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import RecipeForm from './components/RecipeForm';

import RecipeList from './components/RecipeList';

const App = () => {

const [recipes, setRecipes] = useState([]);

const [recipeToEdit, setRecipeToEdit] = useState(null);

useEffect(() => {

fetchRecipes();

}, []);

const fetchRecipes = async () => {

try {

const response = await axios.get('/api/recipes');

setRecipes(response.data);

} catch (error) {

console.error('Error fetching recipes:', error);

}

};

const addRecipe = async (newRecipe) => {

try {

const formData = new FormData();

for (const key in newRecipe) {

formData.append(key, newRecipe[key]);

}

await axios.post('/api/recipes', formData, {

headers: {

'Content-Type': 'multipart/form-data',

},

});

fetchRecipes();

} catch (error) {

console.error('Error adding recipe:', error);

}

};

const deleteRecipe = async (id) => {

try {

await axios.delete(`/api/recipes/${id}`);

fetchRecipes();

} catch (error) {

console.error('Error deleting recipe:', error);

}

};

const updateRecipe = async (updatedRecipe) => {

try {

const formData = new FormData();

for (const key in updatedRecipe) {

formData.append(key, updatedRecipe[key]);

}

await axios.put(`/api/recipes/${updatedRecipe.\_id}`, formData, {

headers: {

'Content-Type': 'multipart/form-data',

},

});

fetchRecipes();

setRecipeToEdit(null);

} catch (error) {

console.error('Error updating recipe:', error);

}

};

return (

<div>

<RecipeForm onAddRecipe={addRecipe} recipeToEdit={recipeToEdit} onUpdateRecipe={updateRecipe}/>

<RecipeList recipes={recipes} onDeleteRecipe={deleteRecipe} onEditRecipe={setRecipeToEdit}/>

</div>

);

};

export default App;

// --- Backend (Node.js with Express.js) ---

// server.js

const express = require('express');

const mongoose = require('mongoose');

const multer = require('multer');

const cors = require('cors');

const Recipe = require('./models/Recipe');

const app = express();

const port = 5000;

app.use(cors());

app.use(express.json());

mongoose.connect('mongodb://localhost:27017/recipe-app', {

useNewUrlParser: true,

useUnifiedTopology: true,

});

const storage = multer.diskStorage({

destination: (req, file, cb) => {

cb(null, 'uploads/');

},

filename: (req, file, cb) => {

cb(null, Date.now() + '-' + file.originalname);

},

});

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

app.get('/api/recipes', async (req, res) => {

try {

const recipes = await Recipe.find();

res.json(recipes);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

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

const recipe = new Recipe({

name: req.body.name,

ingredients: req.body.ingredients,

instructions: req.body.instructions,

category: req.body.category,

prepTime: req.body.prepTime,

image: req.file ? req.file.filename : null,

tags: req.body.tags,

});

try {

const newRecipe = await recipe.save();

res.status(201).json(newRecipe);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

app.delete('/api/recipes/:id', async (req, res) => {

try {

await Recipe.findByIdAndDelete(req.params.id);

res.json({ message: 'Recipe deleted' });

} catch (error) {

res.status(500).json({ message: error.message });

}

});

app.put('/api/recipes/:id', upload.single('image'), async (req, res) => {

try {

const updatedRecipe = await Recipe.findByIdAndUpdate(req.params.id, {

name: req.body.name,

ingredients: req.body.ingredients,

instructions: req.body.instructions,

category: req.body.category,

prepTime: req.body.prepTime,

image: req.file ? req.file.filename : req.body.image,

tags: req.body.tags

}, { new: true });

res.json(updatedRecipe);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

app.listen(port, () => {

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

});

// models/Recipe.js

const mongoose = require('mongoose');

const recipeSchema = new mongoose.Schema({

name: { type: String, required