

**SSN COLLEGE OF ENGINEERING (Autonomous)
Affiliated to Anna University**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Recipe Finder Web Application



Team Members:

1. Ragul B – 3122215001075
2. Sai Rahul – 3122215001090
3. Swathika D - 3122215001114

Course Code: UCS1601

Course Name: Internet Programming Lab

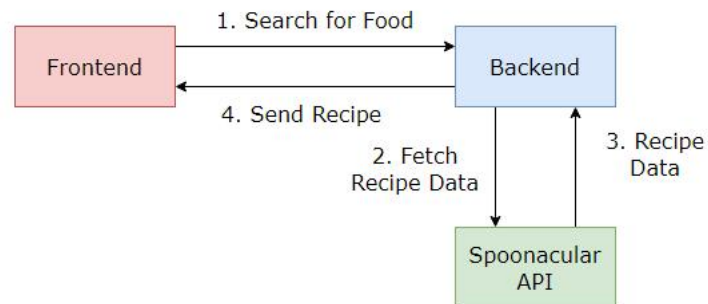
Class: VI Sem - III year

Staff In-charge: Mr. S. Raghavendra Kumar

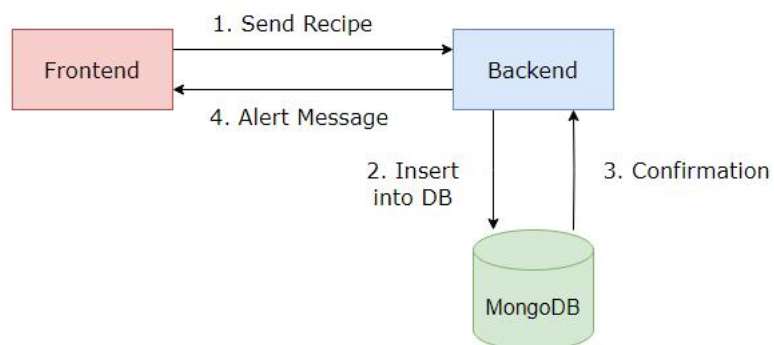
Aim:

Design a full stack Recipe Finder web application which allows users to search for recipes based on the name of the recipe. The app can fetch data from various recipe APIs, display recipes, and provide additional details such as cooking instructions, nutritional information, and user reviews.

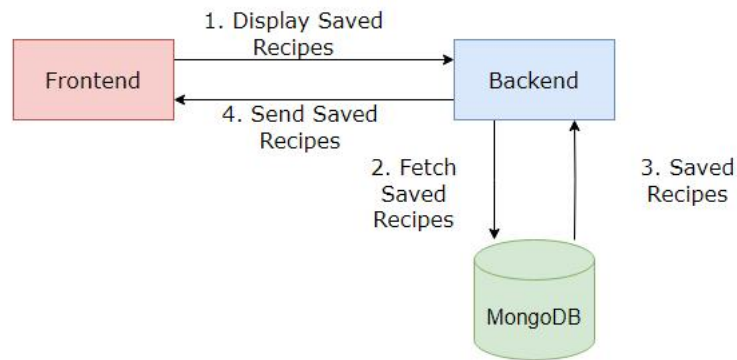
Design Diagram:



Fetching Recipes



Saving recipe



View Saved Recipes

Code:

App.js

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
import Recipe from './Recipe';
import SavedRecipes from './SavedRecipes';
import './App.css';

function App() {
  return (
    <Router>
      <div className="App">
        <header className="App-header">
          <h1>Recipe Finder</h1>
          <nav>
            <Link to="/">Home</Link> | <Link to="/saved">Saved Recipes</Link>
          </nav>
        </header>
        <Routes>
          <Route path="/" element={<Recipe />} />
          <Route path="/saved" element={<SavedRecipes />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

Home.js

```
// components/Home.js
import React from 'react';

const Home = () => {
  return <h1>Welcome to Recipe Finder</h1>;
}

export default Home;
```

index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
  'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
  sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
  monospace;
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
```

```
reportWebVitals();
```

NotFound.js

```
// components/Profile.js
import React from 'react';

const Profile = () => {
  return <h1>Your Profile</h1>;
}

export default Profile;
```

Profile.js

```
// components/Profile.js
import React from 'react';

const Profile = () => {
  return <h1>Your Profile</h1>;
}

export default Profile;
```

Recipe.js

```
// src/components/Recipe.js

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

const Recipe = () => {
  const [search, setSearch] = useState('');
  const [recipes, setRecipes] = useState([]);
  const [sortBy, setSortBy] = useState('strMeal');

  const handleSearchInput = (event) => {
    setSearch(event.target.value);
  };
};
```

```

const fetchRecipes = async () => {
  try {
    const response = await
axios.get(`https://www.themealdb.com/api/json/v1/1/search.php?s=${search}`);
    setRecipes(response.data.meals || []);
  } catch (error) {
    console.error('Error fetching data:', error);
  }
};

const sortRecipes = (key) => {
  setSortKey(key);
  setRecipes(recipes.slice().sort((a, b) => (a[key] > b[key] ? 1 : -1)));
};

const saveRecipe = (recipe) => {
  axios.post('http://localhost:3001/api/recipes', recipe)
    .then(response => {
      alert('Recipe saved!');
    })
    .catch(error => {
      console.error('Error saving recipe:', error);
    });
};

// Function to render ingredients
const renderIngredients = (recipe) => {
  return Object.keys(recipe)
    .filter(key => key.startsWith('strIngredient') && recipe[key])
    .map(key => <li key={key}>{recipe[key]}</li>);
};

return (
  <div>
    <input
      type="text"
      value={search}
      onChange={handleSearchInput}
      placeholder="Enter a dish name..."
    />
    <button onClick={fetchRecipes}>Search</button>
    <button onClick={() => sortRecipes('strMeal')}>Sort by Name</button>
    <button onClick={() => sortRecipes('strCategory')}>Sort by
Category</button>
    {recipes.length > 0 ? recipes.map((recipe) => (
      <div key={recipe.idMeal}>
        <h1>{recipe.strMeal}</h1>

```

```

        <img src={recipe.strMealThumb} alt={recipe.strMeal} style={{width:
'200px', height: '200px', objectFit: 'cover'}} />
        <p><strong>Instructions:</strong>
{recipe.strInstructions.substring(0, 100)}...</p>
        <ul><strong>Ingredients:</strong> {renderIngredients(recipe)}</ul>
        <button onClick={() => saveRecipe(recipe)}>Save</button>
      </div>
    )) : <p>No recipes found. Please try another search.</p>
  </div>
);
}

export default Recipe;

```

RecipeDetails.js

```

// components/RecipeDetails.js
import React from 'react';
import { useParams } from 'react-router-dom';

const RecipeDetails = () => {
  let { id } = useParams();
  return <h1>Details for Recipe {id}</h1>;
}

export default RecipeDetails;

```

RecipeModel.js

```

// src/models/RecipeModel.js
const mongoose = require('mongoose');

const recipeSchema = new mongoose.Schema({
  idMeal: String, // Unique identifier from TheMealDB
  strMeal: String,
  strCategory: String,
  strArea: String,
  strInstructions: String,
  strMealThumb: String,
  strTags: [String],
  ingredients: [String],
});

```

```
const RecipeModel = mongoose.model('Recipe', recipeSchema);

module.exports = RecipeModel;
```

SavedRecipe.js

```
// src/components/SavedRecipes.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const SavedRecipes = () => {
  const [recipes, setRecipes] = useState([]);

  useEffect(() => {
    fetchRecipes();
  }, []);

  const fetchRecipes = async () => {
    try {
      const response = await
axios.get('http://localhost:3001/api/recipes');
      setRecipes(response.data);
    } catch (error) {
      console.error('Failed to fetch recipes:', error);
    }
  };

  return (
    <div>
      <h1>Saved Recipes</h1>
      <div style={{ display: 'flex', flexDirection: 'row', flexWrap:
'wrap' }}>
        {recipes.map(recipe => (
          <div key={recipe.idMeal} style={{ margin: '20px',
textAlign: 'center' }}>
            <h3>{recipe.strMeal}</h3>
            <img src={recipe.strMealThumb} alt={recipe.strMeal}
style={{ width: '200px', height: '200px' }} />
            <p>{recipe.strInstructions.substring(0, 100)}...</p>
          </div>
        ))}
      </div>
    </div>
  );
};
```



```
export default SavedRecipes;
```

server.js

```
// server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const Recipe = require('./RecipeModel'); // Ensure this path matches your
project structure

const app = express();
app.use(cors());
app.use(express.json());

// MongoDB Connection
mongoose.connect('mongodb://localhost:27017/recipeFinder', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
.then(() => console.log('MongoDB connected successfully.'))
.catch(err => console.error('MongoDB connection error:', err));

// POST endpoint to save a recipe
app.post('/api/recipes', async (req, res) => {
  try {
    const existingRecipe = await Recipe.findOne({ idMeal: req.body.idMeal });
    if (existingRecipe) {
      return res.status(409).send('Recipe already exists');
    }

    const recipe = new Recipe(req.body);
    const savedRecipe = await recipe.save();
    res.status(201).json(savedRecipe);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

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

```

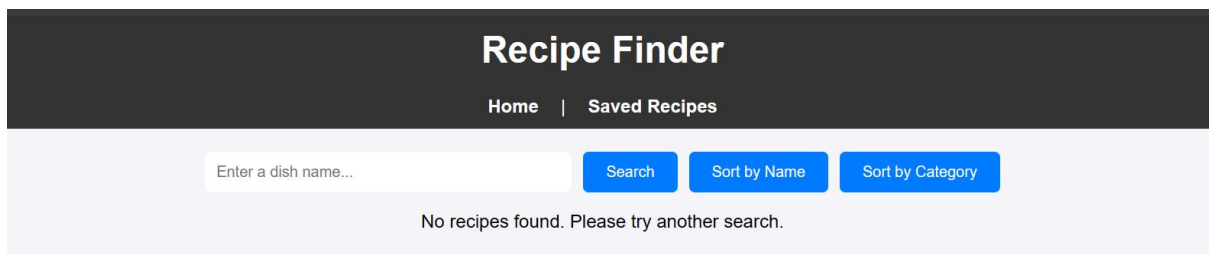
    const recipes = await Recipe.find({});
    res.json(recipes);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});

// GET all recipes
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 });
  }
});

// Server Listening
const PORT = process.env.PORT || 3001;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

```

OUTPUT:




Search

Sort by Name

Sort by Category

Chicken Handi



Instructions: Take a large pot or wok, big enough to cook all the chicken, and heat the oil in it. Once the oil is...

Ingredients:

- Chicken
- Onion
- Tomatoes
- Garlic
- Ginger paste
- Vegetable oil
- Cumin seeds
- Coriander seeds
- Turmeric powder
- Chilli powder
- Green chilli
- Yogurt
- Cream
- fenugreek
- Garam masala
- Salt

Save


Fig 1: Search and Fetch recipes

Search

Sort by Name

Sort by Category

Brown Stew Chicken



Instructions: Squeeze lime over chicken and rub well. Drain off excess lime juice. Combine tomato, scallion, onion...

Ingredients:

- Chicken
- Tomato
- Onions
- Garlic Clove
- Red Pepper
- Carrots
- Lime
- Thyme

Chicken & mushroom Hotpot



Instructions: Heat oven to 200C/180C fan/gas 6. Put the butter in a medium-size saucepan and place over a medium h...

Ingredients:

- Butter
- Onion
- Mushrooms
- Plain Flour
- Chicken Stock Cube
- Nutmeg
- Mustard Powder
- Chicken
- Sweetcorn
- Potatoes
- Butter


Fig 2: Sort by Name

Search

Sort by Name

Sort by Category

Teriyaki Chicken Casserole




Instructions: Preheat oven to 350° F. Spray a 9x13-inch baking pan with non-stick spray. Combine soy sauce, ½ cup...

Ingredients:
soy sauce
water
brown sugar
ground ginger
minced garlic
cornstarch
chicken breasts
stir-fry vegetables
brown rice

Save

Tandoori chicken



Instructions: Mix the lemon juice with the paprika and red onions in a large shallow dish. Slash each chicken thig...

Ingredients:
lemons
paprika
red onions
chicken thighs
vegetable oil
Greek yogurt
cinder

Fig 3: Sort by Category

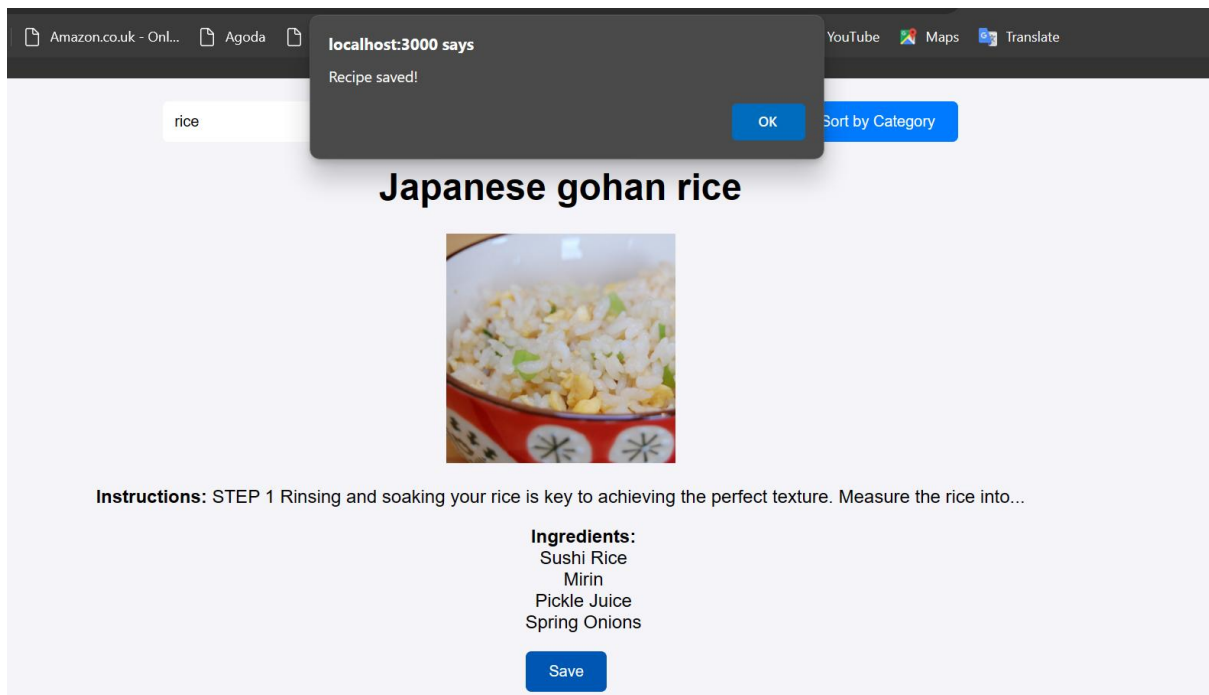


Fig 4: Save Recipe



Fig 5: View Saved Recipes

Learning Outcomes

Searching and Fetching Recipe:

- Design frontend forms for user input.
- Handle and pass user input to backend services.
- Make asynchronous requests to external APIs.

Saving Recipe:

- Implement functionality to save data from frontend to backend.
- Learn about data persistence and storage in databases.

Displaying Saved Recipe:

- Retrieve saved data from backend.
- Dynamically render data on the frontend.
- Understand the flow of data retrieval and display.