Ingredient Explorer(recipe search app)

TEAM MEMBERS:

ROLL NUMBER	NAME
SUJEETHKUMAR M S	22ALR101
SUHIRDHA K S	22ALR100
SUJITH B	22ALR102

INTRODUCTION:

Recipe Radar is an API integration to dynamically load recipe details, ensuring users have access to the most up-to-date and diverse culinary content. Recipes are real-time providing the most current ingredient lists and instructions to follow the recipes. The app fetches the latest recipes and ensure the users to have access across to new and trending dishes. The app uses APIs to analyse user preferences dietary restrictions. Advanced algorithms powered by APIs refine recommendations over time, enhancing user satisfaction and engagement. APIs enable the app to provide detailed cooking instructions, including text, images, and videos.

KEY FEATURES:

1) Recipe Discovery and organization
2) Country List and Regional Dishes
3) It provides an meal planning
4) Provides an cooking guidance
5) It provides an video refernce for cooking
6) Customization and Preferences

REQUIREMENTS

- Front-End Programming

1. CSS (Cascading Style Sheets): CSS is used to define the visual appearance and layout of web pages. It allows developers to apply styles, such as colors, fonts, spacing, and positioning, to HTML elements, creating a visually appealing and consistent user interface.

2. JavaScript: JavaScript is a powerful programming language that adds interactivity and dynamic behavior to web pages. It enables developers to manipulate and modify HTML and CSS, handle user interactions, perform calculations, validate forms, and make asynchronous requests to servers.

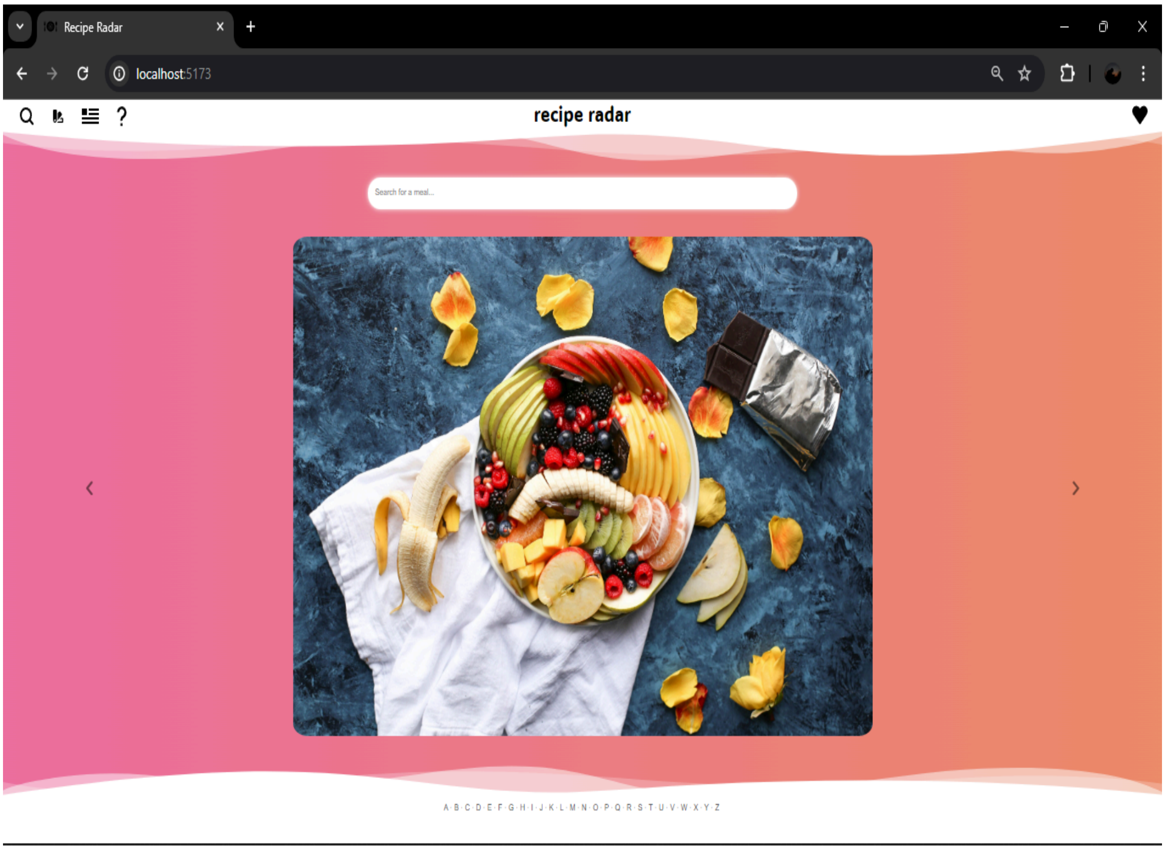
3. React: React is a popular JavaScript library for building user interfaces. It provides a component-based approach to web development, allowing developers to create reusable UI components that update efficiently based on changes in data. React uses a virtual DOM (Document Object Model) to optimize performance and facilitate the building of complex web applications

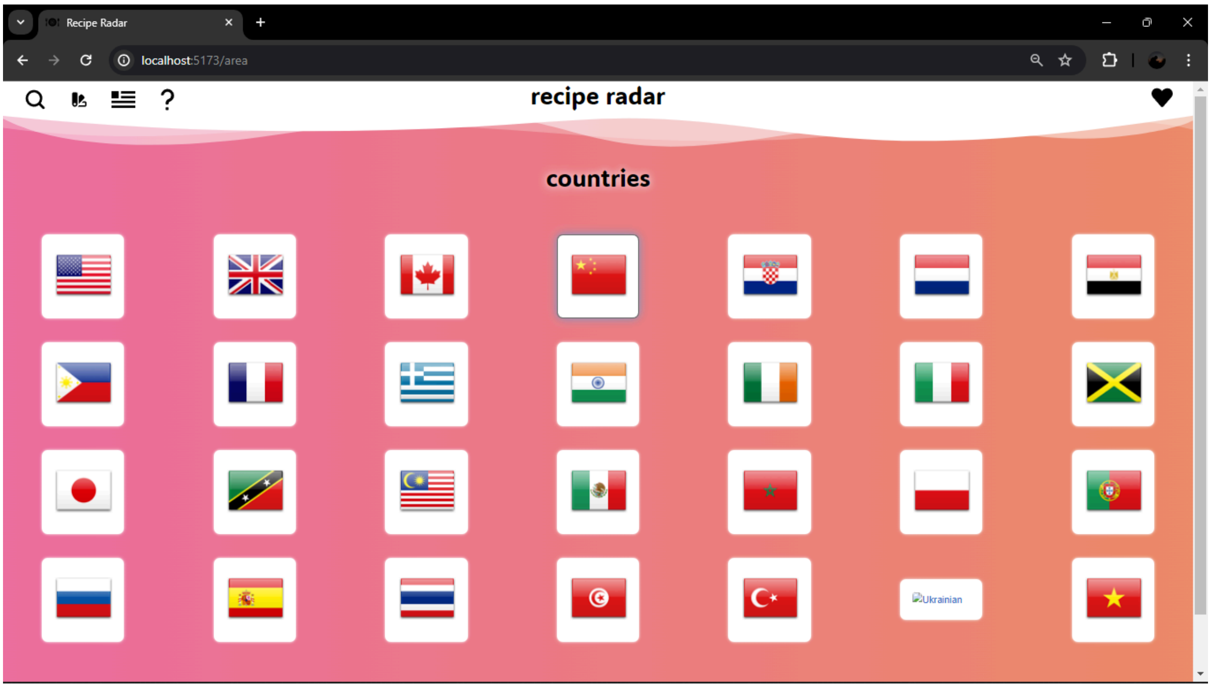
- Back-End Programming

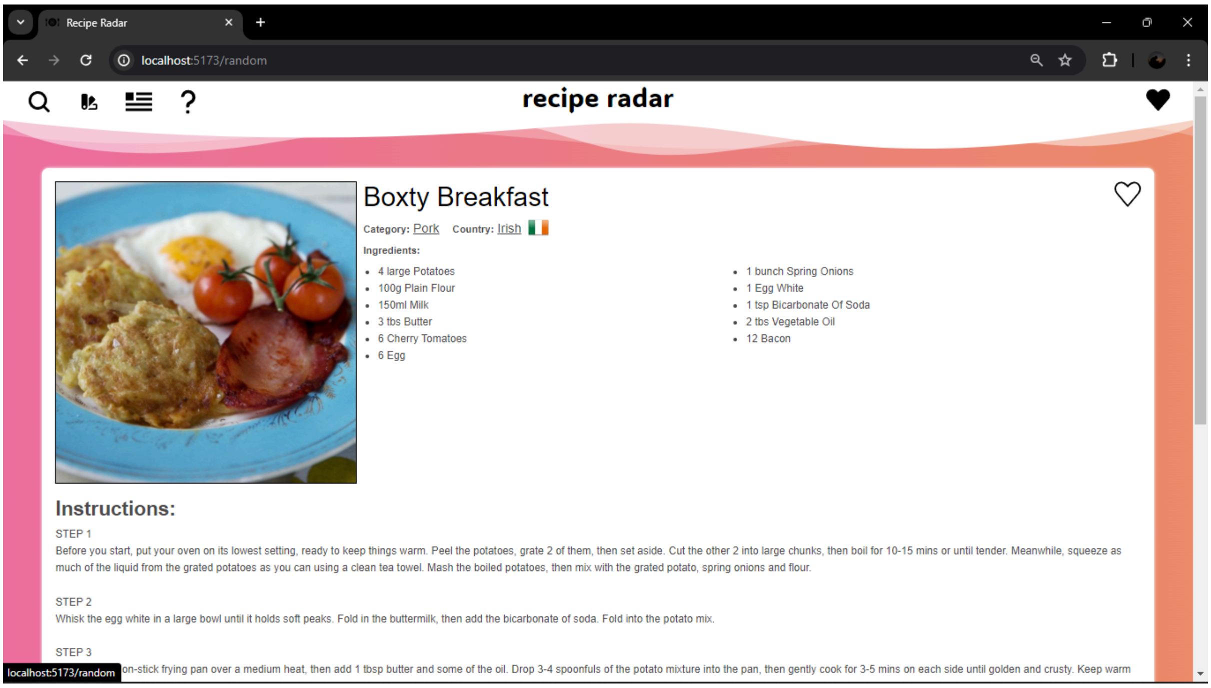
1. APIs (Application Programming Interfaces): Backend developers build APIs that allow communication between the frontend and backend components of a web application. APIs define the protocols and rules for how different software components can interact and exchange data. Commonly used API

standards
include RESTful APIs and GraphQL.

UI DESING:


Recipe Radar
localhost:5173
recipe radar
Search for a meal...
A · B · C · D · E · F · G · H · I · J · K · L · M · N · O · P · Q · R · S · T · U · V · W · X · Y · Z





Boxty Breakfast

Category: Pork **Country:** Irish

Ingredients:

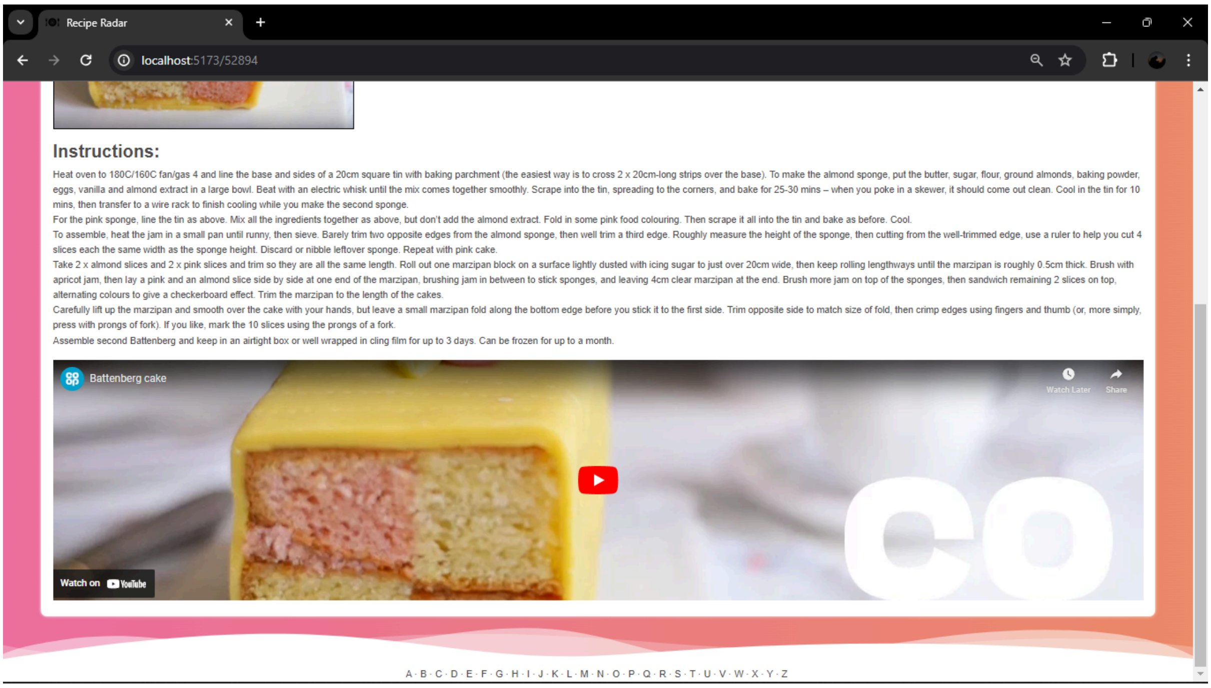
- 4 large Potatoes
- 100g Plain Flour
- 150ml Milk
- 3 tbs Butter
- 6 Cherry Tomatoes
- 6 Egg
- 1 bunch Spring Onions
- 1 Egg White
- 1 tsp Bicarbonate Of Soda
- 2 tbs Vegetable Oil
- 12 Bacon

Instructions:

STEP 1
Before you start, put your oven on its lowest setting, ready to keep things warm. Peel the potatoes, grate 2 of them, then set aside. Cut the other 2 into large chunks, then boil for 10-15 mins or until tender. Meanwhile, squeeze as much of the liquid from the grated potatoes as you can using a clean tea towel. Mash the boiled potatoes, then mix with the grated potato, spring onions and flour.

STEP 2
Whisk the egg white in a large bowl until it holds soft peaks. Fold in the buttermilk, then add the bicarbonate of soda. Fold into the potato mix.

STEP 3
localhost:5173/random on-stick frying pan over a medium heat, then add 1 tbsp butter and some of the oil. Drop 3-4 spoonfuls of the potato mixture into the pan, then gently cook for 3-5 mins on each side until golden and crusty. Keep warm


Recipe Radar
localhost:5173/52894
Battenberg cake
Watch Later
Share
Watch on YouTube
A · B · C · D · E · F · G · H · I · J · K · L · M · N · O · P · Q · R · S · T · U · V · W · X · Y · Z

Instructions:

Heat oven to 180C/160C fan/gas 4 and line the base and sides of a 20cm square tin with baking parchment (the easiest way is to cross 2 x 20cm-long strips over the base). To make the almond sponge, put the butter, sugar, flour, ground almonds, baking powder, eggs, vanilla and almond extract in a large bowl. Beat with an electric whisk until the mix comes together smoothly. Scrape into the tin, spreading to the corners, and bake for 25-30 mins – when you poke in a skewer, it should come out clean. Cool in the tin for 10 mins, then transfer to a wire rack to finish cooling while you make the second sponge.

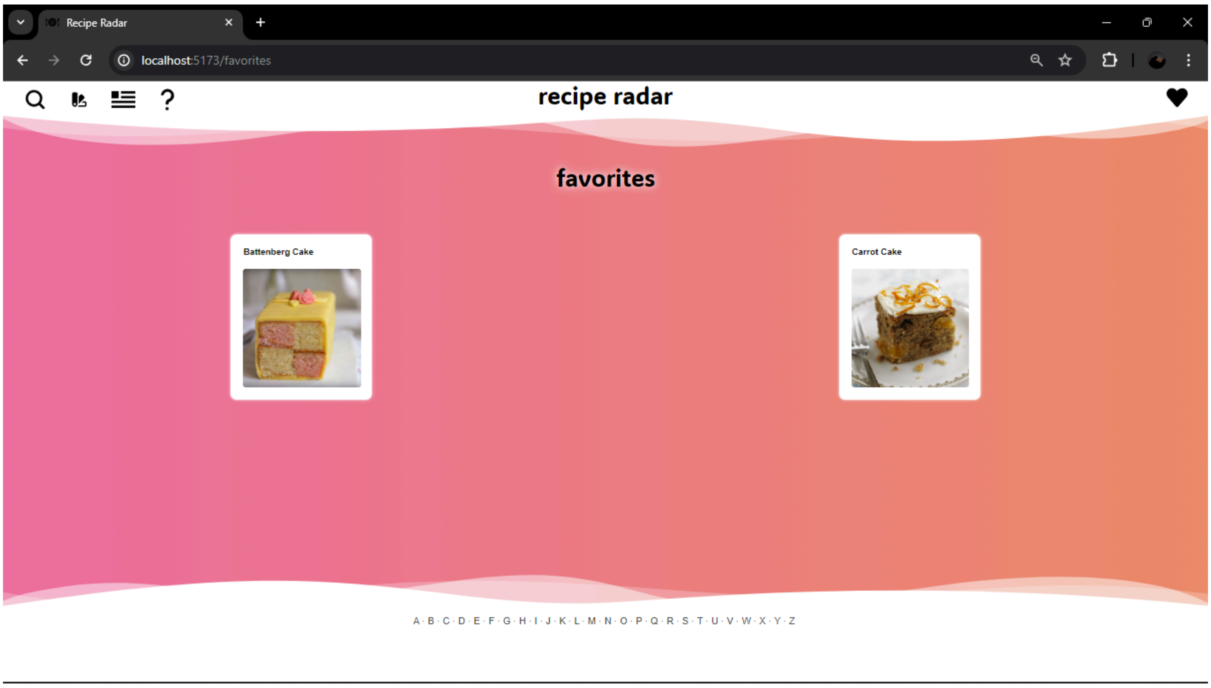
For the pink sponge, line the tin as above. Mix all the ingredients together as above, but don't add the almond extract. Fold in some pink food colouring. Then scrape it all into the tin and bake as before. Cool.

To assemble, heat the jam in a small pan until runny, then sieve. Barely trim two opposite edges from the almond sponge, then well trim a third edge. Roughly measure the height of the sponge, then cutting from the well-trimmed edge, use a ruler to help you cut 4 slices each the same width as the sponge height. Discard or nibble leftover sponge. Repeat with pink cake.

Take 2 x almond slices and 2 x pink slices and trim so they are all the same length. Roll out one marzipan block on a surface lightly dusted with icing sugar to just over 20cm wide, then keep rolling lengthways until the marzipan is roughly 0.5cm thick. Brush with apricot jam, then lay a pink and an almond slice side by side at one end of the marzipan, brushing jam in between to stick sponges, and leaving 4cm clear marzipan at the end. Brush more jam on top of the sponges, then sandwich remaining 2 slices on top, alternating colours to give a checkerboard effect. Trim the marzipan to the length of the cakes.

Carefully lift up the marzipan and smooth over the cake with your hands, but leave a small marzipan fold along the bottom edge before you stick it to the first side. Trim opposite side to match size of fold, then crimp edges using fingers and thumb (or, more simply, press with prongs of fork). If you like, mark the 10 slices using the prongs of a fork.

Assemble second Battenberg and keep in an airtight box or well wrapped in cling film for up to 3 days. Can be frozen for up to a month.


Recipe Radar
localhost:5173/favorites
recipe radar
favorites
Battenberg Cake
Carrot Cake
A · B · C · D · E · F · G · H · I · J · K · L · M · N · O · P · Q · R · S · T · U · V · W · X · Y · Z

SAMPLE CODING:

Index.html

```
<!doctype html>
<html lang="en">
```

```
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=ADLaM+Display&display=swap" rel="stylesheet">
    <link rel="icon" type="image/svg+xml" href="/recipe-radar-icon.svg" />
    <script src="https://kit.fontawesome.com/d377311c2d.js" crossorigin="anonymous"></script>
    <script defer type="module" src="/src/Main.jsx"></script>
    <title>Recipe Radar</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Area.jsx

```
import { Link } from 'react-router-dom';
import { countryCodes } from '../../utils/api';
import './AreaCard.css';

export default function AreaCard ({ strArea }) {
  const url = `https://www.themealdb.com/images/icons/flags/big/64/${countryCodes[strArea]}.png`;

  return (
    <li className='area-card'>
      <Link to={`/area/${strArea}`}>
        <img src={url} alt={strArea} />
      </Link>
    </li>
  );
}
import { Link } from 'react-router-dom';
import './CategoryCard.css';

export default function CategoryCard ({ strCategory, strCategoryThumb })
{
  return (
    <li className='category-card'>
      <Link to={`/category/${strCategory}`}>
        <h1>{strCategory}</h1>
        <img src={`${strCategoryThumb}`}/>
      </Link>
    </li>
  );
}
import { useParams } from 'react-router';
import RecipeCard from '../RecipeCard/RecipeCard';
import '../../utils/spinner.css';
import './CategoryRecipeCardList.css';

export default function CategoryRecipeCardList ({ mealData }) {
  const { category } = useParams();

  function mealDataLoaded() {
    return (
      <>
        <h1 className='label'>{category.toLowerCase()}</h1>
        <ul className='recipe-cards-list'>
          {mealData.map((meal, i) => {
            return <RecipeCard key={i} {...meal}/>
          })}
        </ul>
      </>
    );
  }

  return (
    <section className="category-recipe-cards-container">
      {mealData ? (
        mealDataLoaded()
      ) : (
        <div className='spinner-container'>
          <div
className="lds-spinner"><div></div><div></div><div></div><div></di
v><div></div><div></div><div></div><div></div><div></div><div></
div><div></div><div></div></div>
        </div>
      )}
    </section>
  );
}
```

Categoryrecipecard.jsx

```
import { useParams } from 'react-router';
import RecipeCard from '../RecipeCard/RecipeCard';
import '../../utils/spinner.css';
import './CategoryRecipeCardList.css';

export default function CategoryRecipeCardList ({ mealData }) {
  const { category } = useParams();

  function mealDataLoaded() {
    return (
      <>
        <h1 className='label'>{category.toLowerCase()}</h1>
        <ul className='recipe-cards-list'>
          {mealData.map((meal, i) => {
            return <RecipeCard key={i}  {...meal}/>
          })}
        </ul>
      </>
    );
  }

  return (
    <section className="category-recipe-cards-container">
      {mealData ? (
        mealDataLoaded()
      ) : (
        <div className='spinner-container'>
          <div
className="lds-spinner"><div></div><div></div><div></div><div></di
v><div></div><div></div><div></div><div></div><div></div><div></
div><div></div><div></div></div>
        </div>
      )}
    </section>
  );}
```

Favoritesrecipscard.jsx

```
import RecipeCard from '../RecipeCard/RecipeCard';
import './FavoritesRecipeCardList.css';

export default function FavoritesRecipeCardList ({ favorites }) {
  function favoritesLoaded() {
    return (
      <>
        <h1 className='label'>favorites</h1>
        <ul className='recipe-cards-list'>
          {favorites.map((meal, i) => {
            return <RecipeCard key={i} {...meal} />
          })}
        </ul>
      </>
    );
  }

  return (
    <section className="favorites-recipe-cards-container">
      {favorites.length !== 0 ? (
        favoritesLoaded()
      ) : (
        <div>
          <svg width="800px" height="800px" viewBox="0 0 24 24"
fill="none" xmlns="http://www.w3.org/2000/svg">
            <path d="M12 5.50063C7.50016 0.825464 2 4.27416 2 9.1371C2
14 6.01943 16.5914 8.96173 18.9109C10 19.7294 11 20.5 12 20.5M12
5.50063C16.4998 0.825464 22 4.27416 22 9.1371C22 14 17.9806 16.5914
15.0383 18.9109C14 19.7294 13 20.5 12 20.5M12 5.50063L10.5 8.5L14
11L11 14.5L13 16.5L12 20.5" stroke="#000000d9" strokeWidth="1.5"
strokeLinecap="round" strokeLinejoin="round"/>
          </svg>
          <h1 className='no-fav'>no favorites</h1>
        </div>
      )}
    </section>
  );
}
```

Mealinfo.jsx

```
import { Link } from 'react-router-dom';
import { countryCodes } from '../../utils/api';
import './MealInfo.css';

export default function MealInfo (props) {
  const { favorites, setFavorites, ...restOfProps } = props;
  const url =
`https://www.themealdb.com/images/icons/flags/big/32/${countryCodes[p
rops.strArea]}.png`;
  const indexOfRecipe = props.favorites.findIndex(meal => meal.idMeal
=== props.idMeal);
  const ingredients = [];

  let i = 1;
  while (props[`strIngredient${i}`]) {
    ingredients.push(
      <li className='ingredient-item' key={i}>{props[`strMeasure${i}`]}
{props[`strIngredient${i}`]}</li>
    )
    i++;
  }

  function handleClick() {
    if (indexOfRecipe === -1) {
      const newFavRecipe = {...restOfProps};
      const newFavorites = [newFavRecipe, ...props.favorites];
      localStorage.setItem('favorites', JSON.stringify(newFavorites))
      props.setFavorites(newFavorites)
    } else {
      const favoritesCopy = [...props.favorites];
      favoritesCopy.splice(indexOfRecipe, 1)
      localStorage.setItem('favorites', JSON.stringify([...favoritesCopy]))
      props.setFavorites([...favoritesCopy])
    }
  }

  return (
    <section className='meal-info-component'>
      <div className='meal-img-container'>
        <img src={props.strMealThumb} alt={props.strMeal} />
      </div>
      <div className='meal-info-container'>
        <h1>{props.strMeal}</h1>
        <h2><span>Category:</span> <Link
to={`/category/${props.strCategory}`}
className='meal-info-link'>{props.strCategory}</Link></h2>
        <div className='meal-country-container'>
          <h2><span>Country:</span> <Link to={`/area/${props.strArea}`}
className='meal-info-link'>{props.strArea}</Link></h2>
          <img src={url} alt={props.strArea} />
        </div>
        <h3>Ingredients:</h3>
        <ul className='ingredients-list'>
          {ingredients}
        </ul>
      </div>
      <button className='fav-btn' onClick={handleClick}>
        {indexOfRecipe === -1 ? (
          <svg width="800px" height="800px" viewBox="0 -0.5 32 32"
version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlnsXlink="http://www.w3.org/1999/xlink"
xmlns:sketch="http://www.bohemiancoding.com/sketch/ns">
            <g id="Page-1" stroke="none" strokeWidth="1" fill="none"
fillRule="evenodd" sketch:type="MSPage">
              <g id="Icon-Set" sketch:type="MSLayerGroup" transform="
id="heart-like" sketch:type="MSShapeGroup">
                </path>
              </g>
            </g>
          </svg>
        ) : (
          <svg width="800px" height="800px" viewBox="0 -1 32 32"
version="1.1" xmlns="http://www.w3.org/2000/svg"
xmlnsXlink="http://www.w3.org/1999/xlink"
xmlns:sketch="http://www.bohemiancoding.com/sketch/ns">
            <g id="Page-1" stroke="none" strokeWidth="1" fill="none"
fillRule="evenodd" sketch:type="MSPage">
              <g id="Icon-Set-Filled" sketch:type="MSLayerGroup"
transform="translate(-102.000000, -882.000000)" fill="#000000">
                <path d="" id="heart-like" sketch:type="MSShapeGroup">
                </path>
              </g>
            </g>
          </svg>
        )}
      </button>
    </section>
  );
```

```
}
```

App.jsx

```
import { useState, useEffect } from 'react';
import { Routes, Route } from 'react-router-dom';
import Home from './pages/Home/Home';
import RecipeDetailView from
'./pages/RecipeDetailView/RecipeDetailView';

export default function App() {
  const [favorites, setFavorites] = useState([]);

  useEffect(() => {
    const savedFavorites = localStorage.getItem('favorites');
    if (savedFavorites && savedFavorites !== 'undefined' &&
savedFavorites !== 'null') setFavorites(JSON.parse(savedFavorites))
  }, [])

  return (
    <div className='App'>
      <Header />
      <Routes>
        <Route path='/' element={<Home />} />
        <Route path='/:id' element={<RecipeDetailView
favorites={favorites} setFavorites={setFavorites} />} />
        <Route path='/favorites' element={<Favorites favorites={favorites}
/>} />
        <Route path='/category' element={<Categories />} />
        <Route path='/category/:category' element={<CategoryView />} />
        <Route path='/area' element={<Areas />} />
        <Route path='/area/:area' element={<AreaView />} />
        <Route path='/random' element={<RecipeDetailView
favorites={favorites} setFavorites={setFavorites} />} />
        <Route path='/alphabet/:letter' element={<Alphabet />} />
      </Routes>
      <Footer />
    </div>
  );
}
```

Main.jsx

```
import { StrictMode } from 'react';
import { createRoot } from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom';
import ScrollToTop from './utils/ScrollToTop.jsx';
import App from './App.jsx';
import 'mdb-react-ui-kit/dist/css/mdb.min.css';
import "@fortawesome/fontawesome-free/css/all.min.css";
import './Main.css';

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <BrowserRouter>
      <ScrollToTop />
      <App />
    </BrowserRouter>
  </StrictMode>
)
```

CONCLUSION:

In conclusion,Recipe Radar is an RecipeMaster's integration of a country list and regional dishes allows users to explore and enjoy a diverse array of culinary traditions from around the world. Combined with personalized recommendations, detailed nutritional information, and robust meal planning tools, RecipeMaster provides a comprehensive platform for anyone looking to enhance their cooking skills and culinary knowledge. The app's extensive features ensure that users have everything they need to make delicious and nutritious meals with ease.