

Frontend Development with React.js

Project Title:

CookBook: Your Virtual Kitchen Assistant

TEAM MEMBERS

REGISTER NUMBER	NAME
C24UG104CAP025	S.RAGAVI
C24UG104CAP027	A.RAVEEN
C24UG104CAP034	P.SUMATHI
C24UG104CAP035	R.SWETHA SRI
C24UG104CAP037	K.VIJAYA SRI

Introduction

Welcome to CookBook — Your Virtual Kitchen Assistant. This project aims to create a responsive, accessible, and intelligent recipe web application using React.js. The app helps users discover, save, and cook recipes with step-by-step guidance, meal planning, and offline access.

This document contains the project overview, system flow, detailed feature list, and sample frontend code with editor-style screenshots (VS Code look) to help you understand implementation.

Scenario-Based Intro

Imagine you return home tired and want to cook with ingredients available in your fridge. Open CookBook, enter the ingredients, and the app suggests several quick recipes. Follow the interactive guide to prepare your meal. Share the recipe or add it to your weekly planner for easy shopping list generation.

Target Audience & Problem Statement

Target Audience:

- Food enthusiasts
- Beginners learning to cook
- Health-conscious users
- Busy professionals

Problem Statement:

Many people struggle to decide what to cook with limited ingredients. CookBook solves this by offering ingredient-based suggestions, step-by-step instructions, and planning tools.

Goals and Objectives

1. Provide an intuitive UI for searching and saving recipes.
2. Offer personalized suggestions based on ingredients and preferences.
3. Enable offline access and a meal planner for weekly scheduling.
4. Use modern React.js patterns (hooks, functional components, routing).

Key Features (Detailed)

- Recipe Listings with images, tags, and details.
- Ingredient-based search and filters.
- Favorites (Recipe Book) and Planner integration.
- Step-by-step interactive cooking guide with timers.
- Offline saving and sharing functionality.

System Architecture

Frontend: React.js (Vite or Create React App) with React Router, Axios for API calls, and Bootstrap/Tailwind for styling.

Backend (Mock for development): JSON-Server to simulate REST API (db.json).

Architecture Diagram (conceptual):

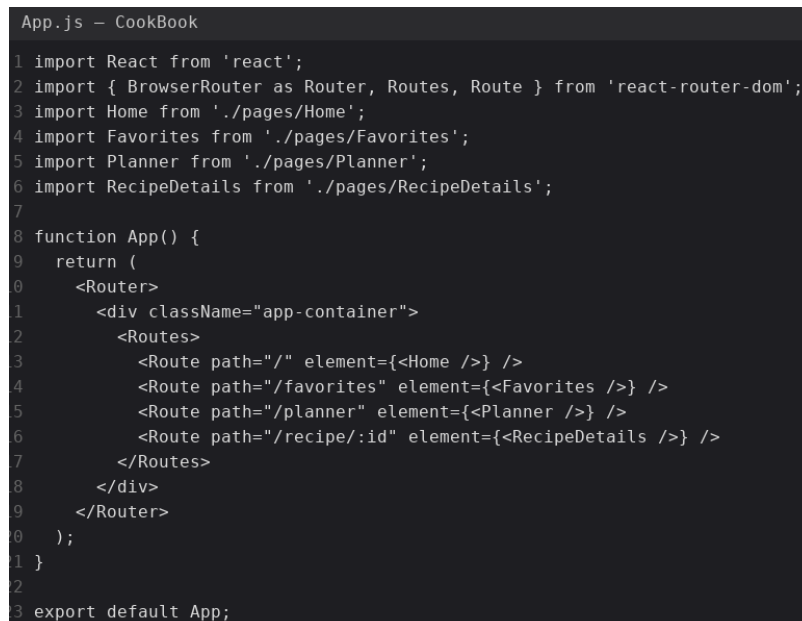
User -> React App -> JSON-Server (recipes)

Coding Examples (React.js) — Editor-style screenshots

Below are sample frontend code files with VS Code-style screenshots. Use these as starting points for your implementation.

App.js

Explanation: This file shows a core part of the frontend. Refer to the screenshot below for exact code.

A screenshot of a code editor showing the App.js file for a project named 'CookBook'. The code defines a function App() that returns a Router configuration. It imports React, React Router components, and several page components (Home, Favorites, Planner, RecipeDetails) from their respective files. The Router is configured with four routes: a home route, a favorites route, a planner route, and a recipe details route with a dynamic ID parameter.

```
App.js — CookBook
1 import React from 'react';
2 import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
3 import Home from './pages/Home';
4 import Favorites from './pages/Favorites';
5 import Planner from './pages/Planner';
6 import RecipeDetails from './pages/RecipeDetails';
7
8 function App() {
9   return (
10     <Router>
11       <div className="app-container">
12         <Routes>
13           <Route path="/" element={<Home />} />
14           <Route path="/favorites" element={<Favorites />} />
15           <Route path="/planner" element={<Planner />} />
16           <Route path="/recipe/:id" element={<RecipeDetails />} />
17         </Routes>
18       </div>
19     </Router>
20   );
21 }
22
23 export default App;
```

App.js: Handles app routing and page structure using React Router v6.

Code :

```
import React from 'react'; import { BrowserRouter as Router, Routes, Route } from 'react-router-dom'; import Home from './pages/Home'; import Favorites from './pages/Favorites'; import Planner from './pages/Planner'; import RecipeDetails from './pages/RecipeDetails'; function App() { return ( <Router> <div className="app-container"> <Routes> <Route path="/" element={<Home />} /> <Route path="/favorites" element={<Favorites />} /> <Route path="/planner" element={<Planner />} /> <Route path="/recipe/:id" element={<RecipeDetails />} /> </Routes> </div> </Router> ); } export default App;
```

RecipeCard.js

Explanation: This file shows a core part of the frontend. Refer to the screenshot below for exact code.


```
RecipeCard.js - CookBook
1 import React from 'react';
2
3 const RecipeCard = ({ recipe, onAddFavorite, onAddToPlanner }) => {
4   return (
5     <div className="card">
6       <img src={recipe.image} alt={recipe.title} className="card-img-top" />
7       <div className="card-body">
8         <h5 className="card-title">{recipe.title}</h5>
9         <p className="card-text">Cuisine: {recipe.cuisine} • Time: {recipe.time} mins</p>
10        <button onClick={() => onAddFavorite(recipe)} className="btn btn-sm btn-outline-danger">♥ Favorite</button>
11        <button onClick={() => onAddToPlanner(recipe)} className="btn btn-sm btn-primary">Add to Planner</button>
12      </div>
13    </div>
14  );
15 };
16
17 export default RecipeCard;
```

RecipeCard.js: Renders individual recipe cards with image, title, and action buttons.

Code :

```
import React from 'react'; const RecipeCard = ({ recipe, onAddFavorite, onAddToPlanner }) => {  
  return ( <div className="card"> <img src={recipe.image} alt={recipe.title}  
    className="card-img-top" /> <div className="card-body"> <h5 className="card-  
    title">{recipe.title}</h5> <p className="card-text">Cuisine: {recipe.cuisine} • Time:  
    {recipe.time} mins</p> <button onClick={() => onAddFavorite(recipe)} className="btn btn-  
    btn-outline-danger">♥ Favorite</button> <button onClick={() => onAddToPlanner(recipe)}  
    className="btn btn-sm btn-primary">Add to Planner</button> </div> </div> ); }; export  
default RecipeCard;  
  
SearchBar.js
```

Explanation: This file shows a core part of the frontend. Refer to the screenshot below for exact code.



```
SearchBar.js - CookBook  
1 import React from 'react';  
2  
3 const SearchBar = ({ searchTerm, setSearchTerm }) => {  
4   return (  
5     <div className="search-bar">  
6       <input  
7         type="text"  
8         placeholder="Search by ingredient, title or cuisine..."  
9         value={searchTerm}  
10        onChange={(e) => setSearchTerm(e.target.value)}  
11      />  
12    </div>  
13  );  
14 };  
15  
16 export default SearchBar;
```

SearchBar.js: Simple controlled component for filtering recipes by text.

Code :

```
import React from 'react'; const SearchBar = ({ searchTerm, setSearchTerm }) => { return (
<div className="search-bar">   <input    type="text"    placeholder="Search by
ingredient, title or cuisine..."    value={searchTerm}    onChange={(e) =>
setSearchTerm(e.target.value)}    />   </div>   ); }; export default SearchBar;

useFetchRecipes.js
```

Explanation: This file shows a core part of the frontend. Refer to the screenshot below for exact code.

```
useFetchRecipes.js - CookBook
1 import { useState, useEffect } from 'react';
2 import axios from 'axios';
3
4 export default function useFetchRecipes() {
5   const [recipes, setRecipes] = useState([]);
6   const [loading, setLoading] = useState(true);
7
8   useEffect(() => {
9     let mounted = true;
10    axios.get('http://localhost:3000/recipes')
11      .then(res => {
12        if (mounted) setRecipes(res.data);
13      })
14      .catch(err => console.error(err))
15      .finally(() => setLoading(false));
16    return () => mounted = false;
17  }, []);
18
19  return { recipes, loading };
20 }
```

useFetchRecipes.js: Custom hook that fetches recipes from the JSON server using Axios.

Code :

```
import { useState, useEffect } from 'react'; import axios from 'axios'; export default function
useFetchRecipes() { const [recipes, setRecipes] = useState([]); const [loading, setLoading] =
useState(true); useEffect(() => { let mounted = true;
axios.get('http://localhost:3000/recipes') .then(res => { if (mounted)
```

```
setRecipes(res.data);    })    .catch(err => console.error(err))    .finally(() =>
setLoading(false));    return () => mounted = false;  }, []);    return { recipes, loading }; }
```

db.json

Explanation: This file shows a core part of the frontend. Refer to the screenshot below for exact code.

```
db.json - CookBook
1 {
2   "recipes": [
3     {
4       "id": 1,
5       "title": "Masala Omelette",
6       "cuisine": "Indian",
7       "time": 10,
8       "ingredients": ["eggs","onion","tomato","green chillies"],
9       "image": "https://example.com/omelette.jpg",
10      "steps": ["Beat eggs", "Chop veggies", "Cook on pan"]
11    },
12    {
13      "id": 2,
14      "title": "Pasta Arrabbiata",
15      "cuisine": "Italian",
16      "time": 25,
17      "ingredients": ["pasta","tomato","garlic","chili flakes"],
18      "image": "https://example.com/pasta.jpg",
19      "steps": ["Boil pasta", "Prepare sauce", "Mix and serve"]
20    }
21  ]
22 }
```

db.json: Sample JSON-Server database containing recipe entries. Use json-server to run locally.

Code :

```
{ "recipes": [ { "id": 1, "title": "Masala Omelette", "cuisine": "Indian",
"time": 10, "ingredients": ["eggs","onion","tomato","green chillies"], "image":
"https://example.com/omelette.jpg", "steps": ["Beat eggs", "Chop veggies", "Cook on pan"]
}, { "id": 2, "title": "Pasta Arrabbiata", "cuisine": "Italian", "time":
25, "ingredients": ["pasta","tomato","garlic","chili flakes"], "image":
```

```
"https://example.com/pasta.jpg",    "steps": ["Boil pasta", "Prepare sauce", "Mix and serve"]  
}  ] }
```

Project Flow and Milestones

Milestone 1 - Setup

- Install Node.js, create React app (Vite or CRA)
- Install React Router, Axios, Bootstrap/Tailwind

Milestone 2 - Core Features

- Recipe listing, search, favorites, planner, recipe details

Milestone 3 - Advanced Features

- Offline support, meal planner, sharing, user accounts

Execution Steps & Deployment

1. Create React app (Vite recommended):
npm create vite@latest cookbook-app --template react
2. Install dependencies:
npm install react-router-dom axios json-server
3. Run JSON server:
npx json-server --watch db.json --port 3000
4. Start dev server:
npm run dev

Deployment: Build and deploy to Netlify / Vercel by running npm run build and connecting repository.

Future Enhancements

- AI-powered recipe suggestions based on dietary history and preferences.
- Voice-guided cooking steps with pause/resume.
- AR overlay for visual guidance while cooking.
- User authentication and personal recipe collections.

Conclusion & Happy Ending

CookBook aims to be the go-to virtual kitchen assistant that makes cooking enjoyable and accessible. 🌟 Bon Appétit! 🌟